

Учреждение образования  
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Кафедра информационных систем и технологий**

# **БАЗЫ ДАННЫХ**

**Методические указания к выполнению контрольной работы  
для студентов специальности 1-40 01 02-03 «Информационные  
системы и технологии (издательско-полиграфический комплекс)»  
заочной формы обучения**

Минск 2012

УДК 004.658:655  
ББК 32.965я73  
Б17

Рассмотрены и рекомендованы к изданию редакционно-издательским советом университета.

С о с т а в и т е л ь  
*В. В. Смелов*

Р е ц е н з е н т  
кандидат технических наук, доцент,  
заведующий кафедрой систем обработки информации  
и полиграфического оборудования учреждения образования  
«Белорусский государственный технологический университет»  
*М. С. Шмаков*

По тематическому плану изданий учебно-методической литературы университета на 2012 год. Поз. 169.

Предназначены для студентов специальности 1-40 01 02-03 «Информационные системы и технологии (издательско-полиграфический комплекс)» заочной формы обучения.

© УО «Белорусский государственный  
технологический университет», 2012

## ПРЕДИСЛОВИЕ

Предлагаемое пособие предназначено для студентов специальности «Информационные системы и технологии (издательско-полиграфический комплекс)» заочной формы обучения, изучающих дисциплину «Базы данных».

По данной дисциплине учебным планом для студентов заочной формы обучения предусматривается контрольная работа. Выполнение и успешная защита контрольной работы являются допуском к сдаче экзамена по этой дисциплине.

Контрольная работа состоит из десяти практических работ, выполняемых студентом самостоятельно. Каждая практическая работа состоит из нескольких заданий. Задания имеют сквозную нумерацию. При выполнении практических работ и заданий следует придерживаться последовательности, предложенной в указаниях, так как формулировка некоторых заданий может опираться на результаты заданий, им предшествующих. Последнее задание каждой практической работы содержит перечень вопросов, на которые студент должен будет ответить при защите этой работы.

Методические указания состоит из десяти разделов, соответствующих практическим работам контрольной работы. Каждый раздел содержит тезисное описание теоретического материала, необходимого для выполнения соответствующей работы. Описание сопровождается ссылками на литературу, рекомендуемую студенту для самостоятельного изучения.

Практические работы подразумевают использование СУБД (система управления базами данных) Microsoft SQL Server 2008 R2. Поэтому для выполнения заданий контрольной работы студенту понадобится компьютер с оперативной памятью не менее 1 ГБ, свободной дисковой памятью не менее 250 МБ и установленной операционной системой Windows версии не ниже XP SP3.

Основная цель контрольной работы – приобретение студентами практических навыков применения языка SQL (Structured Query Language) и его процедурного расширения Transact-SQL для создания объектов базы данных, построения запросов к базе данных и обработки их результатов. СУБД Microsoft SQL Server 2008 рассматривается здесь скорее как инструмент, чем как объект изучения. Поэтому вопросы администрирования, конфигурирования, настройки сервера базы данных в контрольной работе затрагиваются лишь в первой прак-

тической работе и только в том объеме, который понадобится студенту для самостоятельной установки и настройки программного обеспечения, необходимого для выполнения заданий контрольной работы.

Выполнение заданий контрольной работы, за небольшим исключением, сводится к разработке кода на Transact-SQL. Последовательность операторов Transact-SQL часто называют сценариями (или скриптами). Для подготовки сценария, их отладки и выполнения студент должен использовать приложение Microsoft SQL Server Management Studio, входящее в состав СУБД Microsoft SQL Server 2008.

Одно из заданий первой практической работы посвящено установке Microsoft SQL Server 2008 Books Online. Это специальное программное средство для поддержки электронной документации на русском языке. Большинство сведений, необходимых для выполнения заданий практических работ, можно получить из этого источника. В некоторых случаях в тексте будут указываться ссылки на другие источники (в основном это официальные сайты корпорации Microsoft), заслуживающие, по мнению автора, внимания.

Кроме того, в сети Интернет без труда можно найти полезные ресурсы, позволяющих ознакомиться с примерами кода на Transact-SQL. Настоятельно рекомендуется применять запросы в поисковой системе Google следующего формата: *microsoft sql server **смп**ока **зап**оса*. Например, введенная строка *microsoft sql server **create table*** позволяет получить ссылки на ресурсы, содержащие информацию о создании таблиц.

В пособии будут применяться следующие сокращения:

БД – база данных;

СУБД – система управления базой данных;

MSDN – библиотека технической документации подразделения компании Microsoft, ответственного за взаимодействие с пользователями (Microsoft Developer Network);

SQL – Structured Query Language;

MSS – Microsoft SQL Server 2008;

MSS EE – Microsoft SQL Server 2008 Enterprise Edition;

SMS – Microsoft SQL Server Management Studio;

SCM – SQL Server Configuration Manager;

DDL – Data Definition Language;

DML – Data Manipulation Language;

DCL – Data Control Language;

TCL – Transaction Control Language;

T-SQL – Transact-SQL;

BOL – Books Online.

# ВВЕДЕНИЕ

Центральной компонентой информационной системы является база данных. В общем случае база данных представляет собой набор специально организованных и упорядоченных данных, используемых при решении задач, предусмотренных в информационной системе. В современных информационных системах базы данных размещаются на электронных носителях, а для доступа к данным используется комплекс специальных программ, называемых системой управления базой данных (СУБД).

За последние 25 лет архитектура, возможности, состав компонент и прочие свойства СУБД претерпели революционные изменения. Из небольших ограниченных программ с однопользовательским доступом к данным с простой иерархической моделью и рассчитанных на работу с базами данных с размером в несколько мегабайт они превратились в комплексы высокопроизводительных серверов, позволяющих эффективно работать с сотнями терабайтов данных многим пользователям одновременно.

За эти же 25 лет сложился рынок программного обеспечения СУБД. В 2011 году по данным IDC 75% денежных средств, затраченных на покупку СУБД, разделили три компании: Oracle (35%), IBM (30%), Microsoft (10%). В пересчете на количество проданных лицензий (цены СУБД существенно разнятся) среди этих трех компаний лидирует Microsoft (46%). Кроме того, СУБД Microsoft SQL Server 2008 обладает самой низкой стоимостью владения (затраты на поддержку СУБД в работоспособном состоянии) и самую быструю окупаемость (в среднем 3 года).

Независимо от СУБД, современные базы данных обладают рядом одинаковых свойств, которые сложились в результате процесса их эволюции и стандартизации. Практически все современные СУБД предназначены для создания и управления реляционными базами данных. В реляционных базах данных данные организованы в виде таблиц, а для работы с ними используется специальный язык SQL (Structured Query Language – структурный язык запросов). С некоторым приближением реляционную СУБД можно назвать SQL-машиной – автоматом, умеющим исполнять SQL-операторы.

Язык SQL имеет длительную историю стандартизации, однако полного единства этого языка для всех СУБД так и не было достигнуто. Можно лишь говорить о степени соответствия SQL-диалектов

некоторому стандарту. Наиболее устоявшимся, поддерживаемым большинством СУБД стандартом является стандарт SQL, разработанный в 1992 году. Его часто называют SQL92 или SQL2. Считается, что если все запросы приложения к базе данных написаны в соответствии с этим стандартом, то смена СУБД не повлечет за собой необходимости переписывать эти запросы.

Язык SQL включает в себя несколько поименованных подмножеств операторов: DDL (Data Definition Language – язык определения данных), DCL (Data Control Language – язык управления данными), DML (Data Manipulation Language – язык манипулирования данными), TCL (Transaction Control Language – язык управления транзакциями).

DDL – язык, предназначенный для создания, модификации и удаления объектов базы данных. Язык включает три оператора: CREATE – создать объект, ALTER – модифицировать объект, DROP – удалить объект базы данных. С помощью этих операторов создаются, изменяются и удаляются таблицы, индексы, представления, пользователи базы данных, процедуры и другие объекты базы данных.

DCL – язык, предназначенный для управления доступом пользователей базы данных к ее объектам. Язык включает два оператора: GRANT – назначить разрешение, REVOKE – отобрать разрешение на операции с объектом или группой объектов базы данных.

DML – язык, предназначенный для выполнения операций с реляционными таблицами. Язык включает четыре оператора: INSERT – добавить одну или несколько строк в таблицу, DELETE – удалить строки из таблицы, UPDATE – изменить строки таблицы, SELECT – выбрать строки таблицы

TCL – язык, предназначенный для управления транзакциями. Язык включает два основных оператора: COMMIT – зафиксировать изменения в базе данных, ROLLBACK – отказаться от незафиксированных изменений в базе данных (откатить изменения).

Все современные тенденции в области баз данных в полной мере отражены в СУБД Microsoft SQL Server 2008 R2. Изучение возможностей и работа с этой системой управления базами данных даст необходимый набор знаний и навыков, позволяющий охватить семейство технологий, применяемых в современных СУБД.

# **Практическая работа № 1**

## **УСТАНОВКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

### **1.1. Теоретические сведения**

#### **1.1.1. Общие сведения о СУБД Microsoft SQL Server 2008 R2**

Microsoft SQL Server – система управления базами данных (СУБД), разработанная корпорацией Microsoft. Microsoft SQL Server применяется для управления реляционными базами данных средних и крупных размеров. Входит в тройку (Oracle, IBM DB/2, Microsoft SQL Server) самых распространенных СУБД в мире. В настоящее время наиболее популярной версией этого программного продукта является 10.5 или Microsoft SQL Server 2008 R2 (MSS). С историей выпуска версий MSS можно ознакомиться в [2].

MSS имеет широкую линейку вариантов выпуска (поставки покупателю). Выбор варианта зависит от назначения, размера базы данных, оборудования, на котором будет устанавливаться программное обеспечение, а также от финансовых возможностей покупателя. Дальнейший материал пособия ориентирован на вариант Enterprise Edition. Полный перечень вариантов выпуска и их особенности можно найти в [3].

#### **1.1.2. Инсталляция Microsoft Server 2008 Books Online**

Microsoft Server 2008 Books Online (BOL) – программное средство, разработанное корпорацией Microsoft и предназначенное для поддержки электронной документации Microsoft Server 2008 R2. Версию BOL с документацией на русском языке можно скачать с сайта корпорации Microsoft [1]. При выборе скачиваемого варианта BOL следует выбрать документацию на русском языке. Файл, который будет переслан, имеет расширение msi, из чего следует, что это пакет стандартного установщика Windows. Двойной клик мышью на этом файле запустит установщик Windows, который за несколько шагов установит BOL. Общее время установки не превышает 5 минут и не требует специальных знаний или навыков.

После завершения инсталляции запустить BOL можно с помощью меню *Пуск/Все программы/Microsoft SQL Server 2008*. Запуск BOL вызовет появление на мониторе компьютера окна примерно такого вида, как на рис. 1.1.

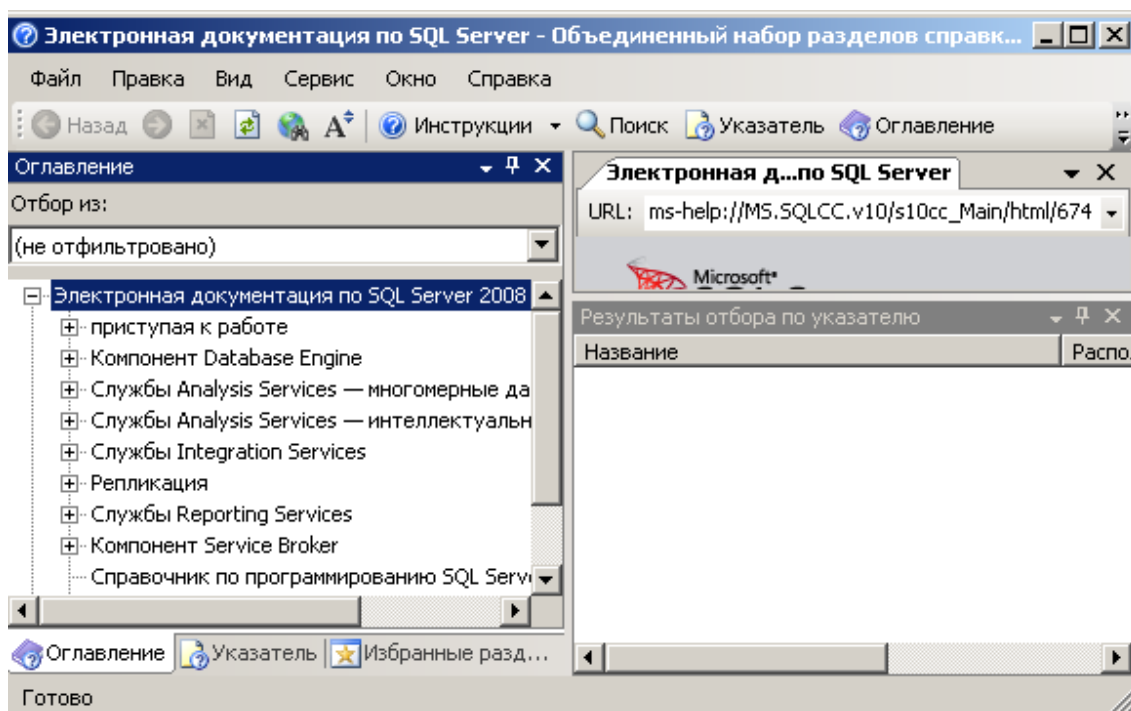


Рис. 1.1. Стартовое окно BOL

Как правило, при установке Microsoft Server 2008 электронная документация (BOL) устанавливается автоматически.

### 1.1.3. Установка Microsoft SQL Server 2008

Для установки MSS необходимо иметь дистрибутивный DVD-диск, содержащий пакет установки MSS, а также компьютер, имеющий не менее 1 ГБ оперативной памяти, не менее 250 МБ свободной дисковой памяти, а также с установленной одной из следующих операционных систем: Windows Server 2003 SP2 / 2003 R2 / 2008 SP2 / 2008 R2, Windows XP SP2 / Vista SP2 / 7. Причем установка MSS EE может быть осуществлена как для 32-, так и на 64-битных вариантах операционных систем.

Запуск пакета установки MSS осуществляется с помощью файла **setup.exe**, который расположен в корневой папке дистрибутивного DVD-диска. Работа пакета установки начинается с окна, представленного на рис. 1.2. Следует обратить внимание на пункт меню **Средство проверки конфигурации**. Вызов его позволяет проверить конфигурацию компьютера, который предназначен для установки MSS. На рис. 1.3 представлено окно, которое получено после выполнения такой проверки. При успешном выполнении напротив каждого проверяемого правила (всего их 13) должна стоять одна из отметок: **Выполнено** или **Неприменимо**.



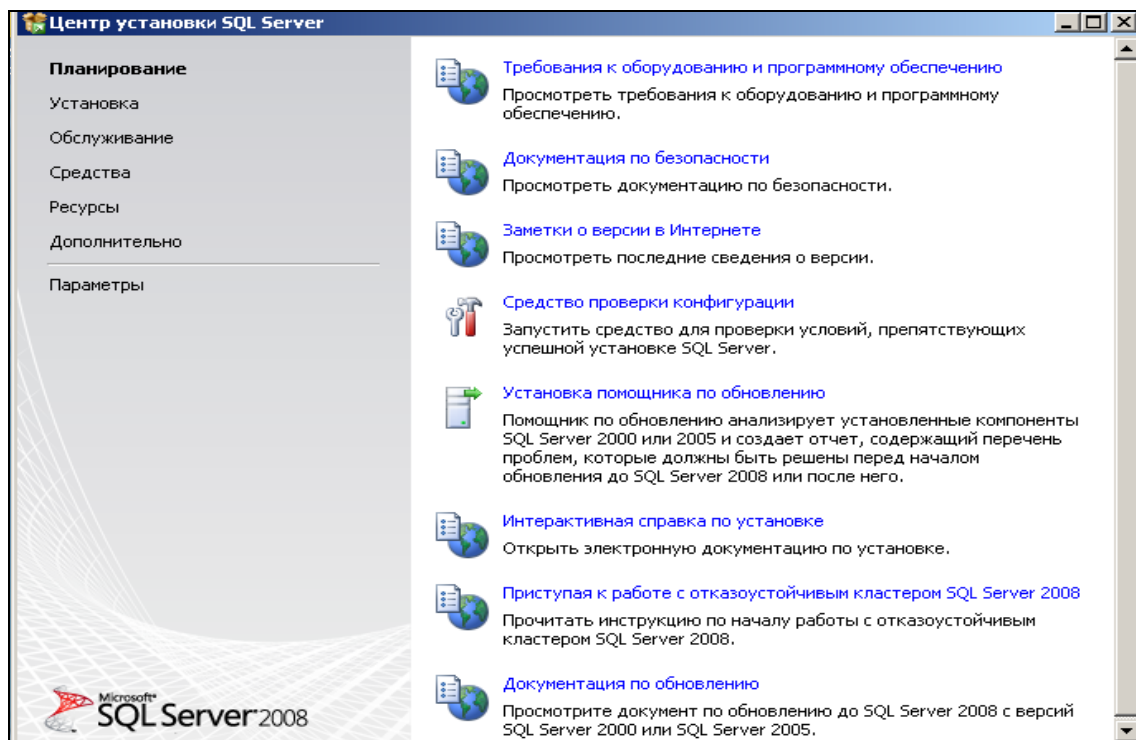


Рис. 1.2. Стартовое окно пакета установки

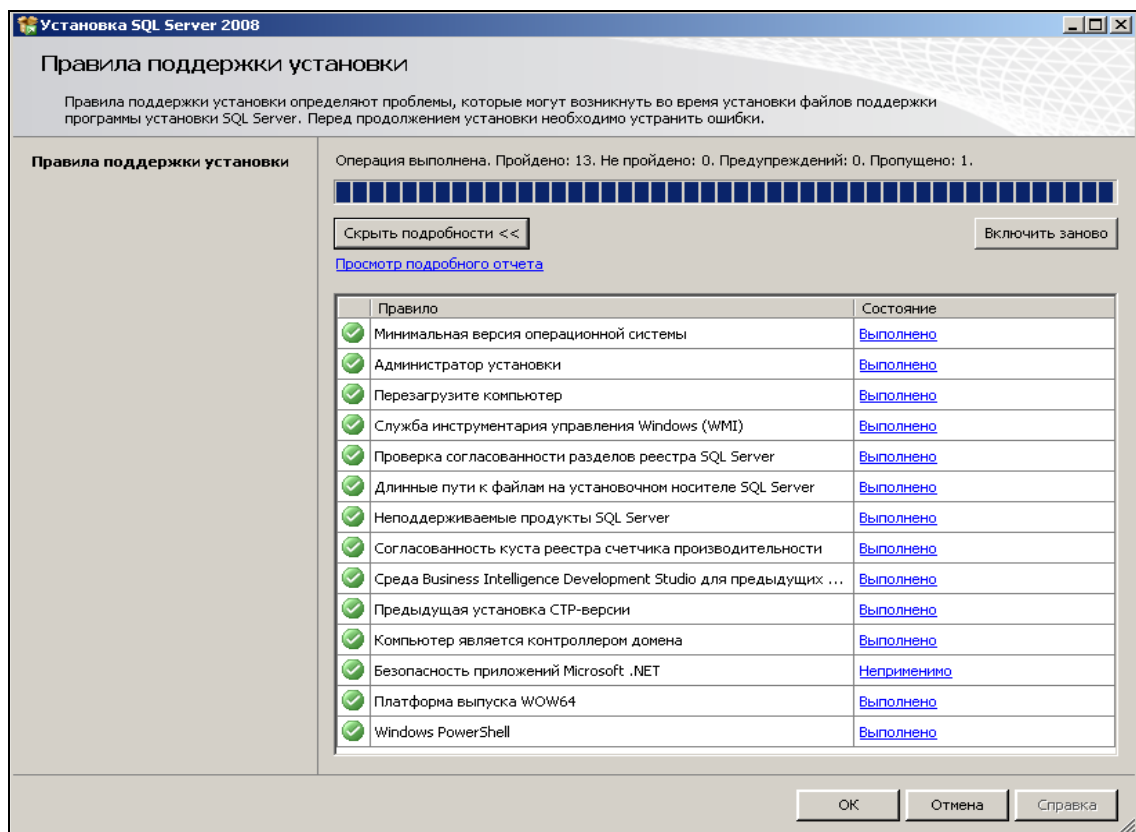


Рис. 1.3. Проверка конфигурации компьютера

Пункт меню **Параметры** позволяет установить тип процессора (x86, x64 или ia64) компьютера, предназначенного для установки MSS, а также корневую папку, в которой будут размещаться устанавливаемые файлы MSS.

Непосредственно установка MSS выполняется в меню **Установка**, в котором предлагается несколько ее вариантов. В нашем случае наиболее подходящим является первый вариант – **Новая установка изолированного сервера**. Процесс установки осуществляется пошагово. На некоторых шагах задаются вопросы, на которые необходимо ответить. Процесс установки может потребовать доступность сети Интернет. В целом установка длится не более 20 минут.

Подробно процесс установки MSS описан в [3, 4].

#### 1.1.4. Настройка сервера и клиента

Для настройки MSS применяется специальное программное средство – **SQL Server Configuration Manager (SCM)**. SCM устанавливается автоматически в процессе установки MSS. Запустить SCM можно после установки MSS с помощью меню **Пуск/Все программы/Microsoft SQL Server 2008/Средства настройки/Диспетчер конфигурации SQL Server**. На рис. 1.4 представлено стартовое окно SCM.

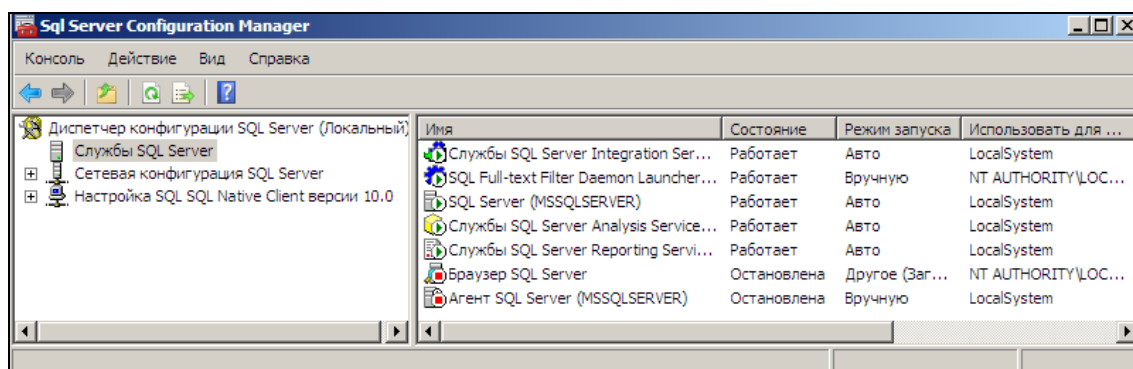


Рис. 1.4. Стартовое окно программы SQL Server Configuration Manager

SCM позволяет конфигурировать, останавливать и запускать службы (компоненты) MSS, настраивать сетевой доступ к серверу, устанавливать параметры сетевого доступа клиентов, создавать псевдонимы сервера и пр.

Более подробно с применением SCM можно ознакомиться в [2].

## 1.2. Задания

### 1.2.1. Задание 1. Установка СУБД Microsoft SQL Server 2008

1. Получите у преподавателя дистрибутивный DVD-диск (или его образ) для установки MSS и создайте его копию.
2. Исследуйте содержимое корневой папки дистрибутивного диска, найдите файл *setup.exe*.
3. Запустите на выполнение файл *setup.exe*.
4. С помощью установочного пакета осуществите предварительную проверку конфигурации компьютера; если выявлены ошибки конфигурации, то устраните их и повторите проверку.
5. Получите скриншот протокола успешной проверки конфигурации и поместите его в отчет о контрольной работе.
6. Установите MSS; все параметры, вводимые во время процесса установки, фиксируйте с помощью скриншотов и отражайте в отчете о контрольной работе.
7. Запустите SCM и убедитесь, что служба SQL Server находится в состоянии «работает».

### 1.2.2. Задание 2. Установка СУБД Microsoft SQL Server 2008 Books Online

1. Выясните: установлен или нет BOL вместе с MSS.
2. Если BOL не установлен, то скачайте [1] и выполните его установочный пакет.
3. Проверьте работоспособность BOL.
4. Проверьте возможность вызова BOL по контексту SMS.

### 1.2.3. Задание 3. Исследование конфигурации сервера

1. С помощью SCM выясните перечень всех служб MSS.
2. С помощью SCM выясните перечень остановленных служб MSS.
3. С помощью SCM выясните местоположение системной базы данных **master** и ее журнала транзакций.
4. Выясните: как обеспечить автоматический старт служб MSS при загрузке операционной системы.
5. Выясните: как запустить и остановить службы MSS с помощью SCM.
6. Результаты исследования отразите в отчете о контрольной работе.

#### **1.2.4. Задание 4. Исследование конфигурации клиента**

1. С помощью SCM выясните перечень сетевых протоколов для доступа к MSS; определите, какие из них отключены.
2. С помощью SCM выясните номер порта TCP по умолчанию, используемый для доступа к MSS.
3. Назначьте TCP/IP-соединению псевдоним SRV-SNF, где S – первая буква вашей фамилии, N – имени, F – отчества.
4. Результаты исследования отразите в отчете о контрольной работе.

#### **1.2.5. Задание 5. Контрольные вопросы**

1. Перечислите параметры MSS, которые вы вводили при установке MSS и поясните их смысл.
2. Поясните понятие «архитектура клиент – сервер».
3. Поясните назначение программы BOL.
4. Поясните назначение программы SCM.
5. Поясните понятие «сервис MSS».
6. Перечислите сетевые протоколы, с помощью которых можно получить доступ к MSS.
7. Назовите стандартный номер порта TCP для доступа к MSS.
8. Для чего используется псевдоним сервера?

## Практическая работа № 2

# СОЗДАНИЕ И УДАЛЕНИЕ БАЗЫ ДАННЫХ

### 2.1. Теоретические сведения

#### 2.1.1. Создание и удаление базы данных с помощью Microsoft SQL Server Management Studio

Один экземпляр MSS может управлять несколькими базами данных. Вся информация о базах данных, управляемых MSS, хранится в системной базе данных **master**.

Для создания базы данных SMS следует воспользоваться контекстным меню *Создать базу данных...*, позволяющим отобразить следующее окно (рис. 2.1).

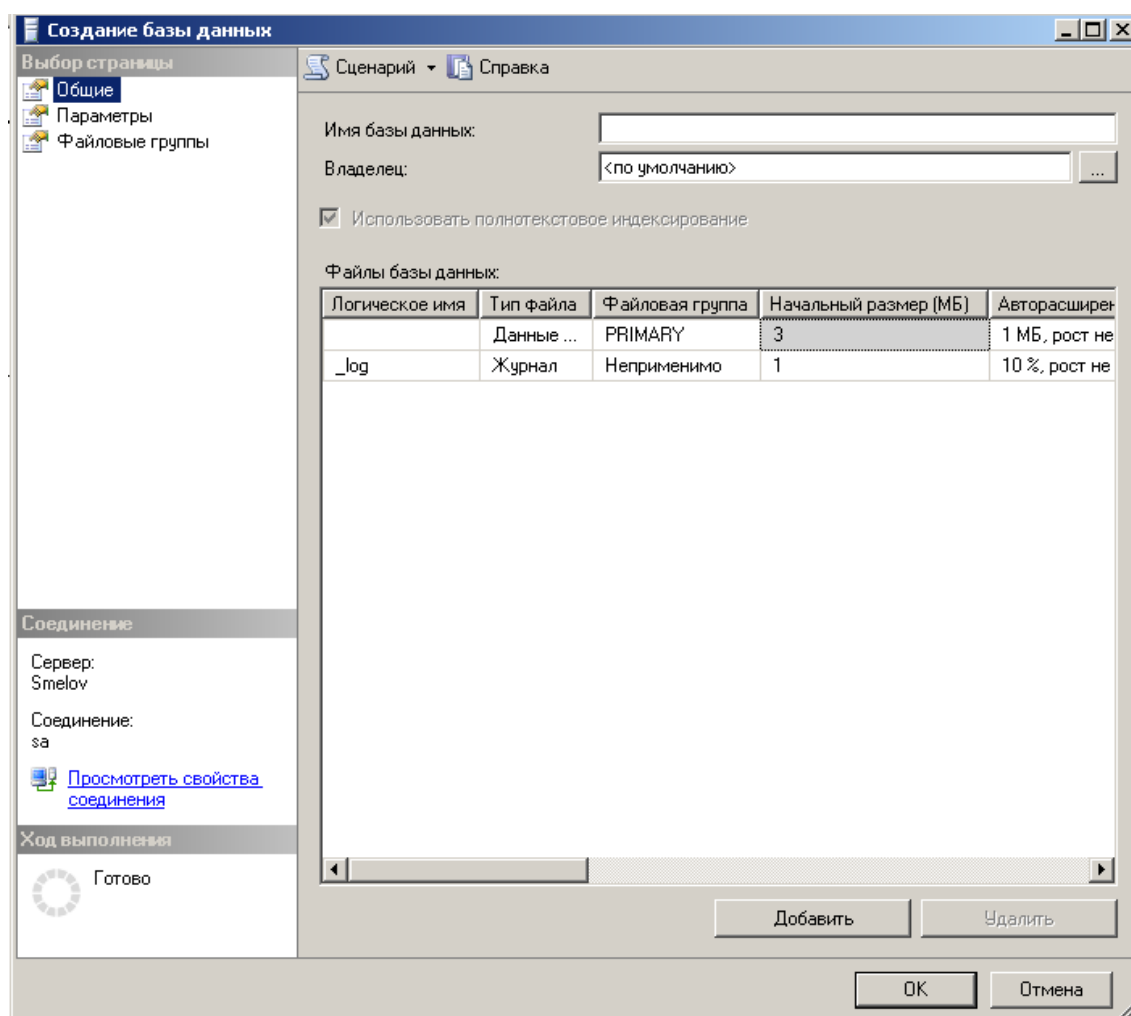


Рис. 2.1. Создание базы данных

Создание базы данных сводится к заполнению трех страниц, которые выбираются с помощью меню в левом верхнем углу окна. Следует обратить внимание на пункт меню **Сценарий**, который позволяет автоматически сформировать SQL-сценарий создания базы данных.

Удалить базу данных можно с помощью контекстного меню, выбрав пункт **Удалить**.

### 2.1.2. Создание и удаление базы данных с помощью T-SQL

Самый простой способ сформировать SQL-сценарий – воспользоваться меню **Сценарий** окна **Создание базы данных SMS** (рис. 2.1) .

Кроме того, для создания SQL-сценария создания базы данных можно воспользоваться контекстным меню SMS **Создать сценарий для базы данных/используя CREATE** (рис. 2.2), с помощью которого можно получить сценарий создания указанной базы данных.

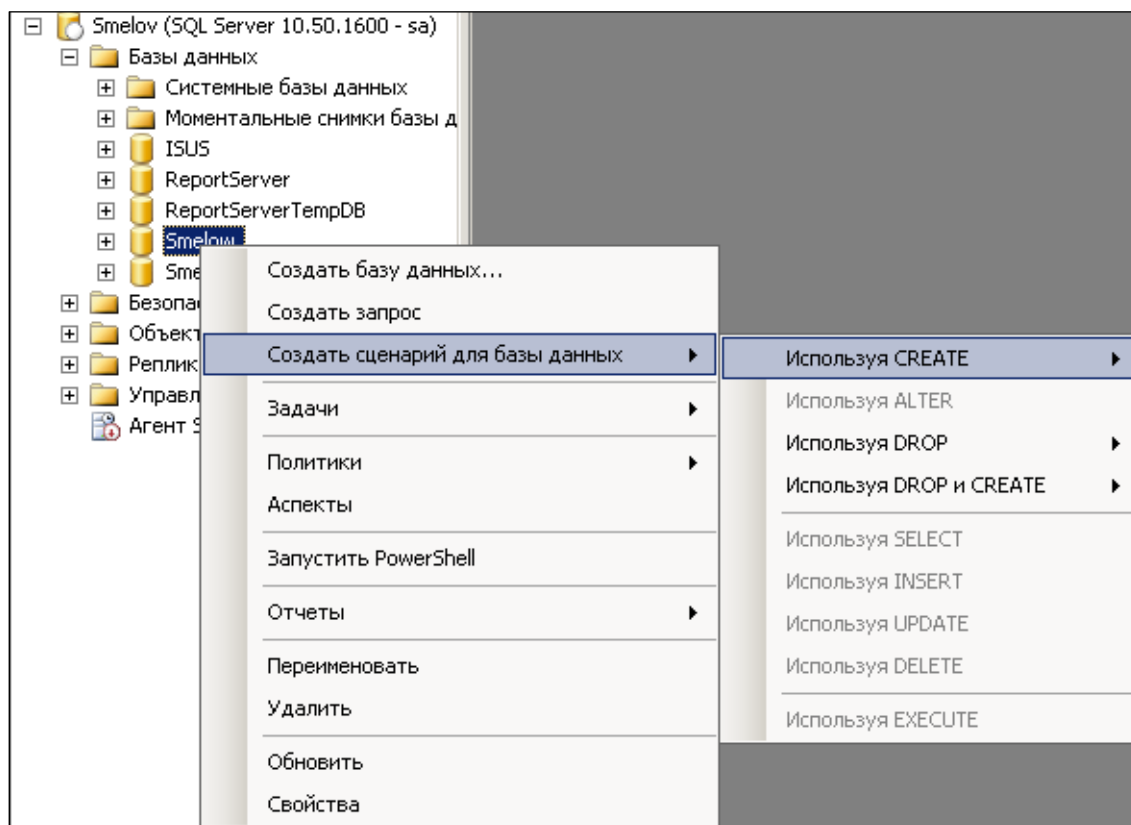


Рис. 2.2. Создание сценария для создания базы данных

Основным оператором, с помощью которого создается база данных, является **CREATE DATABASE**. Кроме того, установка некоторых параметров базы данных может быть выполнена оператором

**ALTER DATABASE.** Удаление базы данных выполняется оператором **DROP DATABASE.** Для знакомства с возможностями и синтаксисом этих операторов можно воспользоваться библиотекой MSDN [5].

### 2.1.3. Отсоединение и присоединение базы данных

Отсоединение и присоединение базы данных используется для переноса базы данных между различными экземплярами MSS. Например, разработанная под управлением сервера **S1** база данных может быть отсоединена от этого сервера, скопирована на какой-нибудь магнитный носитель, восстановлена на магнитный диск другого компьютера и присоединена к серверу **S2**, функционирующему на втором компьютере.

Самый способ отсоединить и присоединить базу данных – это воспользоваться контекстными меню SMS *Задачи/Отсоединить...* (рис. 2.3, 2.4).

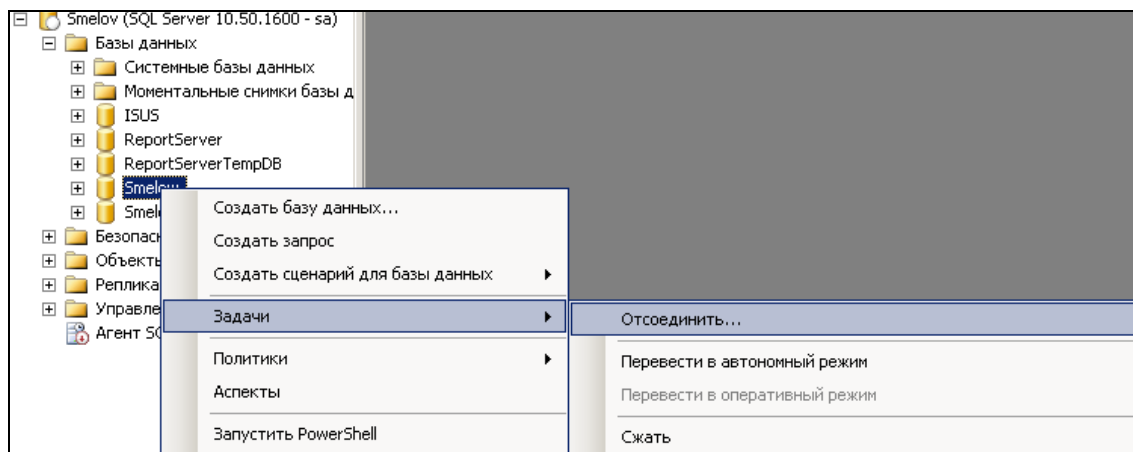


Рис. 2.3. Отсоединение базы данных

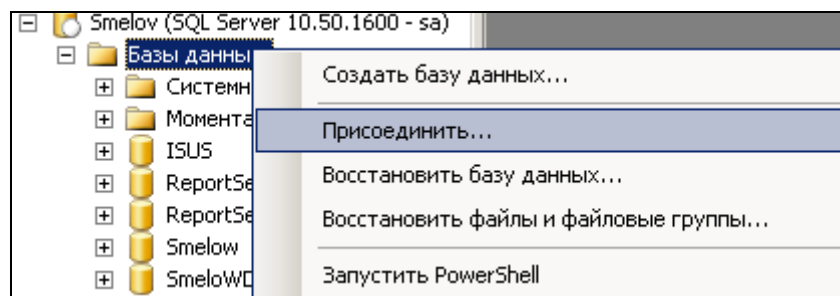


Рис. 2.4. Присоединение базы данных

Кроме того, отсоединение и присоединение базы данных может быть выполнено с помощью T-SQL. Отсоединение базы данных мо-

жет быть выполнено с помощью системной процедуры *sp\_detach\_db*, а присоединение – двумя способами: с помощью системной процедуры *sp\_attach\_db* и с помощью оператора **CREATE DATABASE** с опцией **FOR ATTACH** [5].

## 2.2. Задания

### 2.2.1. Задание 6. Создание базы данных с помощью SMS

1. Создайте базу данных с именем **DBxxxxxx** (где **xxxxxx** – номер зачетной книжки студента), содержащую помимо первичной (**PRIMARY**) еще три файловые группы с именами **G1**, **G2** и **G3**, причем файловая группа **G1** содержит 1 файл, **G2** – 2 файла и **G3** – 3 файла; сформируйте или сохраните сценарий для создания базы данных.

2. Установите значение параметра *модель восстановления* базы данных **Полная**.

3. Установите значение параметра *автоматическое сжатие* базы данных **true**.

4. Установите значение параметра *ограничение доступа* базы данных **MULTI\_USER**.

5. Каждый оператор в сформированном выше сценарии создания базы данных снабдите комментарием, объясняющим назначение оператора.

6. Поместите в отчет о контрольной работе сценарий с комментариями.

7. Удалите базу данных **DBxxxxxx**.

### 2.2.2. Задание 7. Создание базы данных с помощью T-SQL

1. Воспользуйтесь сценарием создания базы данных, сформированном в предыдущем задании.

2. Поменяйте значение параметра *модель восстановления* базы данных на **С неполным протоколированием**.

3. Поменяйте значение параметра *автоматическое сжатие* базы данных на **false**.

4. Сохраните новый сценарий и поместите его в отчет о контрольной работе.

5. Выполните откорректированный сценарий создания базы данных.

6. С помощью SMS убедитесь, что база данных **DBxxxxxx** создана.



### 2.2.3. Задание 8. Отсоединение и присоединение базы данных с помощью SMS

1. Отобразите в отчете о контрольной работе скриншот окна SMS *Свойства базы данных*, отображающего месторасположение файлов базы данных **DBxxxxxx**, созданной в предыдущем задании.
2. Отсоедините базу данных **DBxxxxxx**.
3. Переместите файлы отсоединенной базы данных на другое место (в другие папки жесткого диска).
4. Присоедините базу данных **DBxxxxxx** с учетом того, что файлы базы данных были перемещены.
5. Отобразите в отчете о контрольной работе скриншот окна SMS *Свойства базы данных*, отображающего месторасположение файлов присоединенной базы данных **DBxxxxxx**.

### 2.2.4. Задание 9. Контрольные вопросы

1. Перечислите типы файлов, которые используются базой данных, и поясните их назначение.
2. Что такое файловая группа и для чего они используются?
3. Для чего используется первичная (PRIMARY) файловая группа?
4. Поясните смысл параметра *Модель восстановления*.
5. Поясните смысл параметра *ANSI NULL по умолчанию*.
6. Поясните смысл параметра *Включено прерывание при делении на ноль*.
7. Поясните смысл параметра *База данных доступна только для чтения*.
8. Поясните смысл параметра *Ограничения доступа*.
9. С помощью какого оператора T-SQL создается база данных?
10. С помощью какого оператора T-SQL могут быть изменены параметры базы данных?
11. С помощью какого оператора T-SQL может быть удалена база данных?
12. Для чего применяется отсоединение и присоединение базы данных?
13. Перечислите способы отсоединения базы данных.
14. Перечислите свойства присоединения базы данных.

## **Практическая работа № 3**

# **СОЗДАНИЕ И УДАЛЕНИЕ ТАБЛИЦ**

### **3.1. Теоретические сведения**

#### **3.1.1. Таблицы и ограничения целостности базы данных**

Основными объектами реляционной базы данных являются таблицы. При проектировании базы данных принимается решение о перечне таблиц, их структурах (состав столбцов, их типы и наименования), а также об ограничениях, которые накладываются на данные, хранящиеся в столбцах таблицы. Совокупность структур таблиц и ограничений базы данных называется *логической схемой*. С принципами проектирования баз данных можно ознакомиться в [6].

Совокупность ограничений на данные, хранящиеся в столбцах таблиц, называются *ограничениями целостности базы данных*. Каждое такое ограничение поименовано. Имя ограничению задается автоматически, по умолчанию, или указывается специально при создании или модификации таблицы. Поименованное ограничение называют *констрейнтом*.

MSS поддерживает следующие виды ограничений: *PRIMARY KEY*, *FOREIGN KEY*, *UNIQUE*, *CHECK*, *NULL*, *NOT NULL*, *DEFAULT*. Все ограничения задаются при создании таблицы и в некоторых случаях могут быть отменены или изменены при модификации таблиц. С описанием ограничений и принципами их применения можно ознакомиться в [5].

#### **3.1.2. Создание и удаление таблиц с помощью SMS**

Для создания таблицы можно воспользоваться контекстным меню SMS (рис. 3.1), позволяющим создать вкладку (рис. 3.2), предназначенную для описания свойств столбцов таблицы и ограничений.

Для удаления существующей таблицы тоже можно использовать контекстное меню, установив предварительно курсор на удаляемую таблицу. Следует помнить, что если между таблицами установлено отношение *PRIMARY KEY / FOREIGN KEY*, то удаление таблицы с *PRIMARY KEY* не возможно до тех пор, пока существует ссылающаяся на нее таблица с *FOREIGN KEY*.

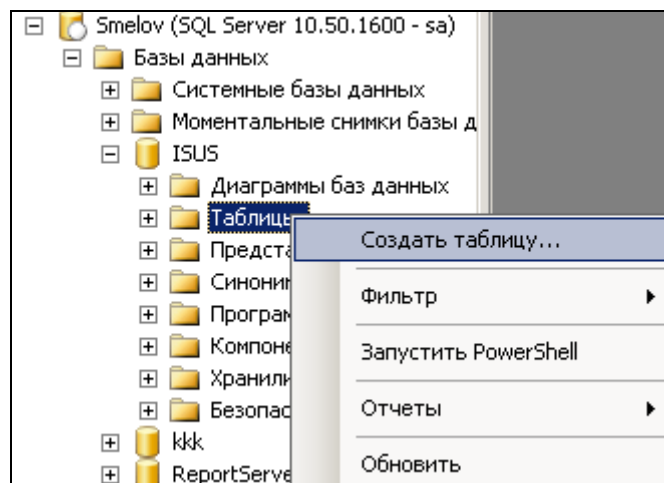


Рис. 3.1. Создание таблиц

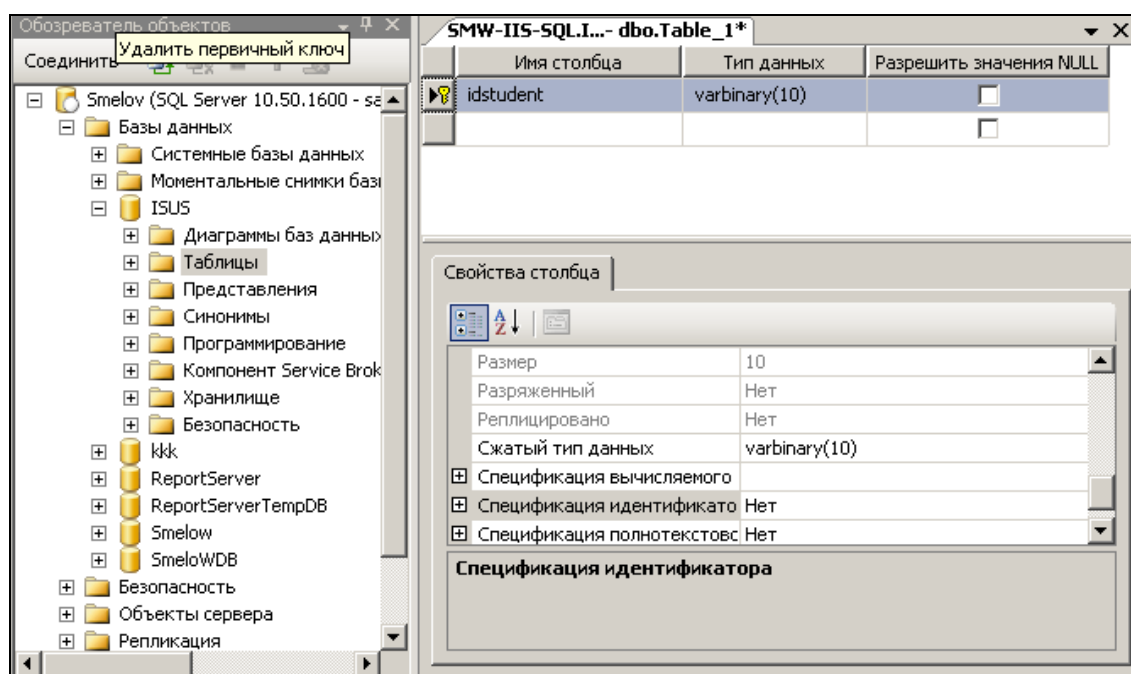


Рис. 3.2. Описание свойств столбцов таблицы

### 3.1.3. Создание и удаление таблиц с помощью T-SQL

Создать таблицу базы данных можно с помощью SQL-оператора **CREATE TABLE**.

Если таблица уже существует, то сценарий для создания таблицы может быть получен с помощью контекстного меню в SMS (рис. 3.3).

Аналогично может быть построен сценарий для удаления таблицы с помощью оператора **DROP**.

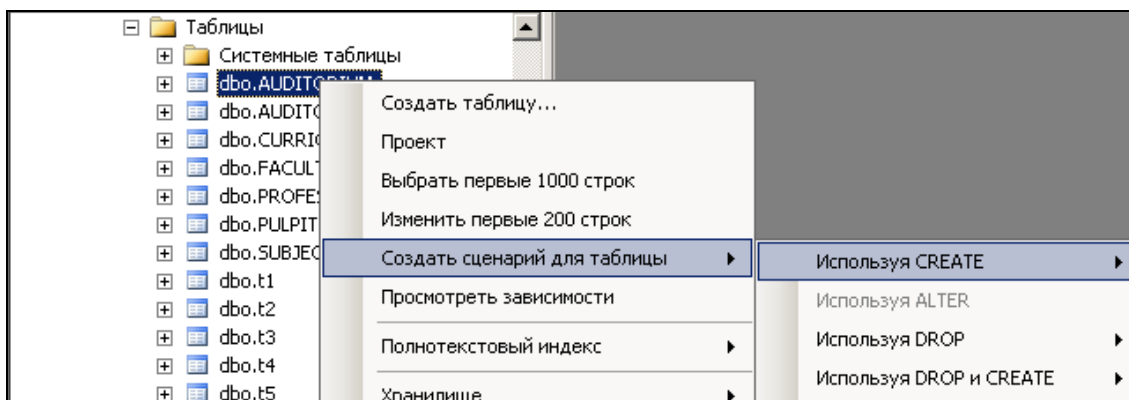


Рис. 3.3. Формирование сценария для создания таблицы

Модификацию структуры таблицы и ее ограничений можно выполнить с помощью оператора **ALTER TABLE**. С полным синтаксисом операторов **CREATE**, **DROP** и **ALTER TABLE** можно ознакомиться в [5].

## 3.2. Задания

### 3.2.1. Задание 10. Создание таблиц с помощью SMS

1. Ознакомьтесь с диаграммой логической схемы базы данных, представленной на рис. 3.4; база данных описывает структуру вуза и содержит информацию о факультетах (таблица **FACULTY**), спиральностях, по которым проводится обучение (**PROFESSION**) на факультетах, кафедрах (**PULPIT**), преподавателях (**TEACHER**), работающих на этих кафедрах, и дисциплинах (**SUBJECT**), закрепленных за кафедрами; кроме того, в базе данных содержится информация об учебных аудиториях вуза (**AUDITORIUM**), которые классифицируются по типам (**AUDITORIUM\_TYPE**).

2. Ознакомьтесь с табл. 3.1, содержащей описание таблиц, изображенных на диаграмме (рис. 3.4); в таблице используются сокращенные обозначения для ограничений: PK – PRIMARY KEY, FK – FOREIGN KEY.

3. Создайте в базе данных **DBxxxxxx** с помощью SMS таблицы **AUDITORIUM\_TYPE** и **AUDITORIUM** в соответствии с описанием в табл. 3.1; таблицы должны находиться в файловой группе **G1**.

### 3.2.2. Задание 11. Создание таблиц помощью T-SQL

1. Разработайте сценарий для создания таблиц **FACULTY**, **PROFESSION**, **PULPIT**, **TEACHER** и **SUBJECT** в соответствии с описа-

нием в табл. 3.1 в базе данных *DBxxxxxx*; таблицы *FACULTY*, *PROFESSION*, *PULPIT* должны располагаться в файловой группе *G2*, таблицы *TEACHER* и *SUBJECT* – в файловой группе *G3*.

2. С помощью SMS создайте диаграмму базы данных *DBxxxxxx* и поместите ее в отчет о контрольной работе.

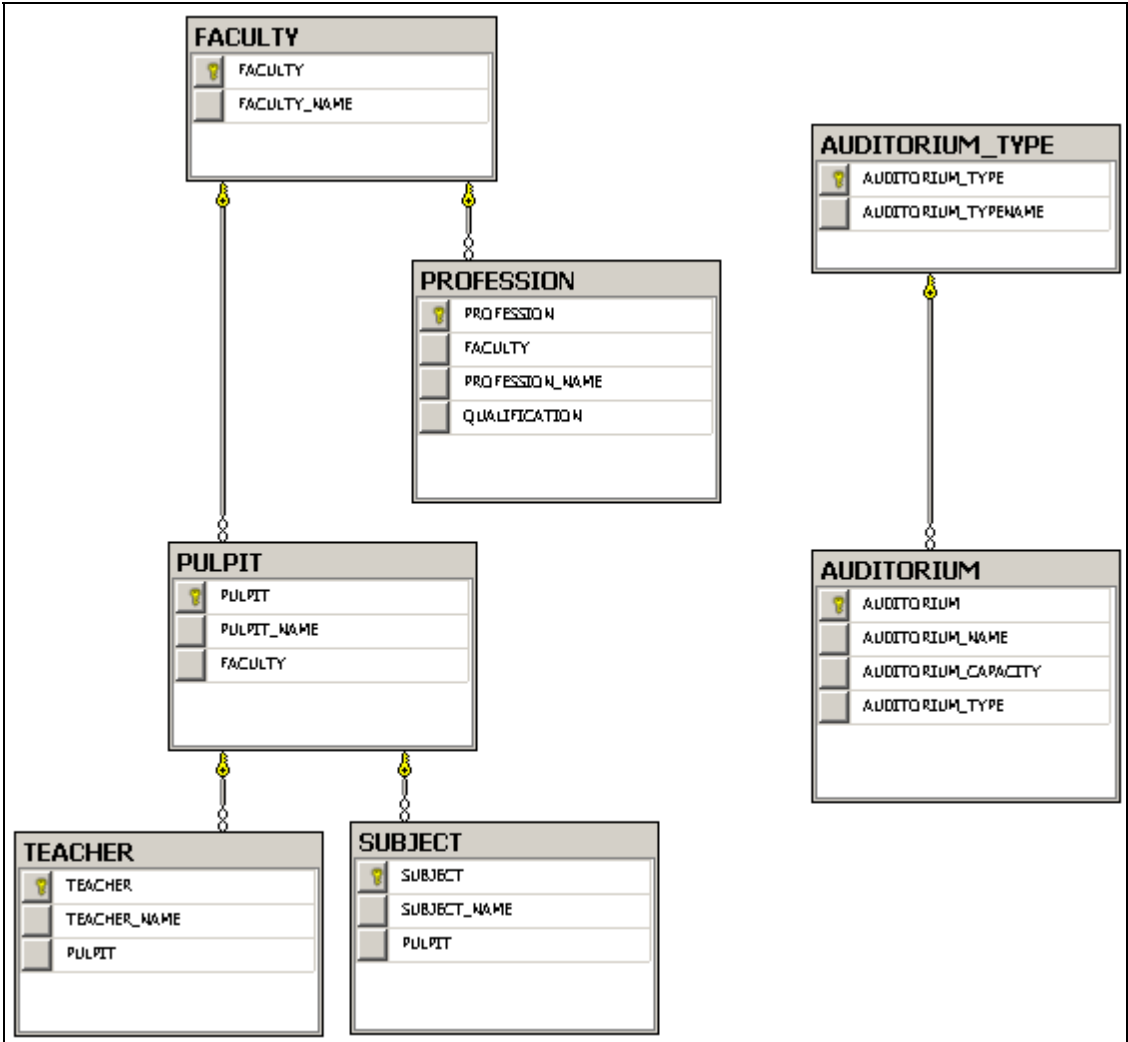


Рис. 3.4. Диаграмма логической схемы базы данных

Таблица 3.1

Перечень таблиц и их структура

Имя таблицы	Столбцы таблицы	Наименование и свойства столбцов
FACULTY	Факультеты	
	FACULTY	код факультета, PK, char(10), not null
	FACULTY_NAME	наименование факультета, varchar(50), default '???'

Имя таблицы	Столбцы таблицы	Наименование и свойства столбцов
PROFESSION	Специальности	
	PROFESSION	код специальности, PK, char(20), not null
	FACULTY	код факультета, FK(FACULTY), char(10), not null
	PROFESSION_NAME	наименование специальности, varchar(100), null
	QUALIFICATION	квалификация, varchar(50), null
PULPIT	Кафедры	
	PULPIT	код кафедры, PK, char(20), not null
	PULPIT_NAME	наименование кафедры, varchar(100), null
	FACULTY	код факультета, FK(FACULTY), char(10), not null
TEACHER	Преподаватели	
	TEACHER	код преподавателя, PK, char(10), not null
	TEACHER_NAME	фамилия, имя, отчество преподавателя, varchar(100), null
	PULPIT	код кафедры, FK(PULPIT), char(10), not null
SUBJECT	Дисциплины	
	SUBJECT	код дисциплины, PK, char(10), not null
	SUBJECT_NAME	наименование дисциплины, varchar(100), null, unique
	PULPIT	код кафедры, FK(PULPIT), char(20), not null
AUDITORIUM_TYPE	Типы учебных аудиторий	
	AUDITORIUM_TYPE	код типа аудитории, PK, char(10), not null
	AUDITORIUM_TYPE-NAME	наименование типа аудитории, varchar(30), null
AUDITORIUM	Учебные аудитории	
	AUDITORIUM	код аудитории, PK, char(20), not null
	AUDITORIUM_TYPE	код типа аудитории, FK(AUDITORIUM_TYPE), char(10), not null
	AUDITORIUM_CAPACITY	вместимость аудитории, int, default 1, check between 1 and 300
	AUDITORIUM_NAME	наименование аудитории, varchar(50), null

### 3.2.3. Задание 12. Контрольные вопросы

1. Что такое *логическая схема базы данных*?
2. Что такое *ограничения целостности базы данных*?
3. Что такое *констрейнт*?
4. Перечислите виды ограничений для столбцов и поясните их смысл.
5. Перечислите типы данных для хранения символьной информации, поддерживаемых MSS, и поясните принцип их применения.
6. Перечислите типы данных для хранения числовой информации, поддерживаемых MSS, и поясните принцип их применения.
7. Перечислите типы данных для хранения даты и времени, поддерживаемых MSS, и поясните принцип их применения.
8. Объясните назначение и принцип применения свойства *INDENTITY* столбца таблицы.
9. Что такое *вычисляемые столбцы*?
10. Перечислите операторы T-SQL, с помощью которых можно создать, удалить и модифицировать таблицу базы данных.

## Практическая работа № 4

### ПРИМЕНЕНИЕ SQL DML

#### 4.1. Теоретические сведения

##### 4.1.1. Оператор INSERT

Оператор *INSERT* предназначен для добавления новых строк в таблицу. На рис. 4.1 представлен пример применения оператора *INSERT* для добавления одной новой строки в таблицу *TEACHER*.

```
use ISUS          -- контекст базы данных ISUS

insert into TEACHER(TEACHER, TEACHER_NAME, PULPIT)
values('ГРБЮВ', 'Горбунова Юлия Владимировна', 'ИСИТ');
```

Рис. 4.1. Пример применения оператора INSERT

Существуют другие формы записи оператора *INSERT*, позволяющие одним оператором добавлять в таблицу более одной строки. Пример, представленный на рис. 4.2, добавляет в таблицу *TTT* подмножество строк таблицы *TEACHER*.

```
use ISUS          -- контекст базы данных ISUS

insert into TTT(T, P) select TEACHER, PULPIT from TEACHER where PULPIT = 'ИСИТ';
```

Рис. 4.2. Пример применения оператора INSERT  
для ввода более одной строки

С полным описанием возможностей оператора INSERT можно ознакомиться в [5].

##### 4.1.2. Оператор SELECT

Оператор *SELECT* предназначен для поиска и выбора строк из таблицы. Это самый мощный и наиболее часто используемый DML-оператор. Оператор позволяет создавать сложные конструкции для поиска данных из одной или нескольких таблиц, гибкого выбора под-



множества строк, группировки и сортировки данных, а также для выполнения различных вычислений.

На рис. 4.3 приведено несколько примеров применения оператора **SELECT**.

```
use ISUS          -- контекст базы данных ISUS

select * from FACULTY                                --1

select PROFESSION, PROFESSION_NAME from PROFESSION order by PROFESSION --2

select TEACHER, PULPIT from TEACHER where PULPIT = 'ИСИТ' --3

select p.FACULTY, t.TEACHER_NAME from TEACHER t join PULPIT p
      on t.PULPIT = p.PULPIT                        --4

select AUDITORIUM_TYPE, sum(AUDITORIUM_CAPACITY) ss from AUDITORIUM
      group by AUDITORIUM_TYPE                      --5

select at.AUDITORIUM_TYPE from AUDITORIUM_TYPE at
      where exists (select * from AUDITORIUM aa
                    where aa.AUDITORIUM_CAPACITY > 50
                    AND aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE) --6

select AUDITORIUM_TYPE, sum(AUDITORIUM_CAPACITY) from AUDITORIUM --7
      group by AUDITORIUM_TYPE
      compute sum(sum(AUDITORIUM_CAPACITY))
```

Рис. 4.3. Примеры применения оператора SELECT

В тех случаях, когда результаты двух SELECT-запросов (два множества строк) имеют одинаковую структуру (однотипный набор столбцов), можно использовать команды **UNION**, **INTERSECT**, **EXCEPT** для выполнения операций объединения, пересечения и разности между множествами строк (рис. 4.4).

```
use ISUS

select TEACHER_NAME from TEACHER where TEACHER like 'А%' -- 1
union
select TEACHER_NAME from TEACHER where TEACHER like 'Б%'

select PULPIT from PULPIT -- 2
except
select distinct PULPIT from TEACHER

select FACULTY from PULPIT -- 3
intersect
select FACULTY from PROFESSION
```

Рис. 4.4. Примеры применения команд UNION, INTERSECT, EXCEPT

С полным описанием возможностей оператора **SELECT** можно ознакомиться в [5].

#### 4.1.3. Оператор DELETE

Оператор **DELETE** предназначен для удаления строк из заданной таблицы. Для выбора подмножества удаляемых строк используются практически такие же конструкции, как и в операторе **SELECT**. Следует обратить внимание: удаление строк осуществляется только из одной таблицы.

На рис. 4.5 приведено несколько примеров применения оператора **DELETE**.

```
delete TEACHER                                --1
delete TEACHER where TEACHER = 'CMJB'         --2
delete TEACHER from TEACHER t join PULPIT p on t.PULPIT = p.PULPIT --3
        where not exists (
                select * from PROFESSION f
                where f.FACULTY = p.FACULTY and
                f.PROFESSION = '1-40 01 02'
        )
```

Рис. 4.5. Примеры применения оператора DELETE

Для выполнения удаления всех строк таблицы часто используют специальный оператор **TRUNCATE**.

С полным описанием возможностей оператора **DELETE** и особенностями выполнения оператора **TRUNCATE** можно ознакомиться в [5].

#### 4.1.4. Оператор UPDATE

Оператор **UPDATE** предназначен для изменения значений столбцов в строках заданной таблицы. Для выбора подмножества изменяемых строк используются практически такие же конструкции, как и в операторах **SELECT** и **DELETE**.

Следует обратить внимание: изменение строк осуществляется только в одной таблице.

На рис. 4.6 приведено несколько примеров применения оператора **UPDATE**.

С полным описанием возможностей оператора **UPDATE** можно ознакомиться в [5].

```

update  TEACHER set  TEACHER_NAME = TEACHER + '-' + TEACHER_NAME;      --1

update TEACHER set  TEACHER_NAME = 'Иванов Сергей Борисович',          --2
               TEACHER = 'ИБН'
where  TEACHER = 'СМИБ'

update TEACHER set  TEACHER_NAME = TEACHER_NAME + ' +'                  --3
from  TEACHER t join PULPIT p on t.PULPIT = p.PULPIT
where exists (
               select * from PROFESSION f
               where f.FACULTY = p.FACULTY and
                     f.PROFESSION = '1-40 01 02'
            )

```

Рис. 4.6. Примеры применения оператора UPDATE

## 4.2. Задания

### 4.2.1. Задание 13. Применение оператора INSERT

1. Используя информацию раздела «Факультеты и кафедры» сайта [7], ознакомьтесь с перечнем таблиц в табл. 3.1.
2. Ознакомьтесь с сайтом Белорусского государственного технологического университета [7].
3. Используя [7], заполните с помощью операторов *INSERT* таблицы *FACULTY* и *PULPIT*.
4. Используя раздел «Абитуриентам» сайта [7], заполните с помощью операторов *INSERT* таблицу *PROFESSION*.
5. Используя разделы «Профессорско-преподавательский состав кафедры» сайта [7], заполните с помощью операторов *INSERT* таблицу *TEACHER*.
6. Используя разделы «Учебная работа» сайта [7], заполните с помощью операторов *INSERT* таблицу *SUBJECT*.
7. Заполните с помощью операторов *INSERT* таблицы *AUDITORIUM\_TYPE* и *AUDITORIUM*, используя содержимое табл. 4.1 и 4.2.
8. Приведите пример применения конструкции *INSERT ... SELECT* (см. рис. 4.2) для ввода информации о перечне всех лекционных аудиторий из таблицы *AUDITORIUM* в другую, предварительно созданную, таблицу *LK*, состоящую из одного столбца с именем *AUD*.
9. С помощью оператора *SELECT* выведите содержимое всех заполненных таблиц и поместите в отчет о контрольной работе.

Таблица 4.1

**Перечень типов аудиторий**

Тип аудитории	Наименование типа аудитории
?	???
ЛБ-Х	Химическая лаборатория
ЛБ-СК	Специализированный компьютерный класс
ЛК	Лекционная аудитория
ЛК-К	Лекционная аудитория с компьютерами
ЛБ-К	Компьютерный класс

Таблица 4.2

**Перечень аудиторий**

Код	Наименование	Вместимость	Тип аудитории
?	???	0	ЛК
103-4	103-4	80	ЛК
105-4	105-4	80	ЛК
107-4	107-4	80	ЛК
110-4	110-4	80	ЛК
111-4	111-4	50	ЛК
114-4	114-4	70	ЛК-К
131-4	131-4	20	ЛБ-К
132-4	132-4	80	ЛК
137-4	137-4	50	ЛК
200-3a	200-3a	140	ЛК
229-4	229-4	80	ЛК
236-1	236-1	80	ЛК-К
2Б-4	02Б-4	100	ЛК
301-1	301-1	20	ЛБ-СК
304-4	304-4	20	ЛБ-К
305-4	305-4	110	ЛК-К
309-1	309-1	22	ЛБ-СК
310a-1	310a-1	20	ЛБ-К
313-1	313-1	85	ЛК
314-4	314-4	80	ЛК
320-4	320-4	80	ЛК
324-1	324-1	90	ЛК
408-2	408-2	80	ЛК
410-3a	410-3a	20	ЛБ-Х
413-1	413-1	20	ЛБ-К
415-1	322-3	22	ЛБ-Х
423-1	423-1	20	ЛБ-К
429-4	429-4	80	ЛК
5-1	5-1	60	ЛК

#### 4.2.2. Задание 14. Применение оператора **SELECT**

1. Для выполнения заданий используйте заполненные в предыдущем задании таблицы; все пункты задания выполняйте с помощью оператора **SELECT**.

2. Получите перечень всех факультетов университета.

3. Получите перечень всех лекционных аудиторий с вместимостью более 80.

4. Получите перечень всех специальностей, в названии квалификации которых содержится слово **химик**.

5. Получите перечень наименований специальностей, соответствующих квалификаций, а также наименование факультета, на котором обучаются студенты данной специальности; перечень должен быть отсортирован по коду факультета.

6. Получите перечень всех дисциплин, которые преподаются на факультете **ИДП**.

7. Получите перечень преподавателей с указанием наименования факультета, на котором работают данные преподаватели; перечень должен быть отсортирован в алфавитном порядке фамилий.

8. Получите перечень наименований всех факультетов с указанием количества кафедр на каждом факультете; перечень должен быть отсортирован в порядке убывания количества кафедр (первая строка перечня должна содержать наименование факультета с наибольшим количеством кафедр).

9. Получите перечень наименований всех факультетов с указанием количества преподавателей на каждом факультете; перечень должен быть отсортирован в порядке возрастания количества преподавателей (первая строка перечня должна содержать наименование факультета с наименьшим количеством преподавателей).

10. Вычислите суммарную, среднюю, максимальную и минимальную вместимости всех аудиторий.

11. Вычислите суммарную, среднюю, максимальную и минимальную вместимости аудиторий для каждого типа аудиторий.

12. Приведите пример конструкции **SELECT ... WITH CUBE** и объясните результат.

13. Приведите пример конструкции **SELECT ... WITH ROLLUP** и объясните результат.

14. Приведите пример конструкции **SELECT ... HAVING** и объясните результат.

15. Приведите пример конструкции **SELECT ... COMPUTE** и объясните результат.

#### 4.2.3. Задание 15. Применение оператора UPDATE

1. Для выполнения заданий используйте заполненные в предыдущем задании таблицы; все пункты задания выполняйте с помощью оператора **UPDATE**.

2. Измените наименование аудитории **415-1** так, чтобы оно совпало с кодом.

3. Увеличьте вместимость всех аудиторий на 10%.

4. Установите вместимость 85 для аудитории **324-1**.

5. Установите наименование **амфитеатр** для аудитории **200-3a**.

6. Уменьшите вместимость на 5 всех лекционных аудиторий.

7. Увеличьте вместимость на 3 всех лекционных аудиторий с компьютерами, код которых заканчивается символами **-4**.

#### 4.2.4. Задание 16. Применение оператора DELETE

1. Для выполнения заданий используйте заполненные в предыдущем задании таблицы; все пункты задания (если специально не оговорен другой способ) выполняйте с помощью оператора **DELETE**; после выполнения задания восстановите содержимое таблиц **AUDITORIUM\_TYPE** и **AUDITORIUM** для соответствия их табл. 4.1 и 4.2.

2. Удалите информацию обо всех аудиториях, вместимость которых превышает 100.

3. Удалите информацию обо всех аудиториях, вместимость которых превышает 20, но меньше 60 (в секции **WHERE** используйте операцию **BETWEEN**).

4. Удалите информацию обо всех аудиториях, вместимость которых превышает на 10% среднюю вместимость всех аудиторий.

5. Удалите с помощью оператора **TRUNCATE** все строки таблицы **LK**, созданной в задании 13; с помощью оператора **DROP** удалите таблицу **LK**.

#### 4.2.5. Задание 17. Контрольные вопросы

1. Перечислите все группы операторов языка SQL.

2. Расшифруйте аббревиатуру DML на английском языке и переведите ее на русский.

3. Перечислите все операторы, входящие в группу DML.

4. Объясните назначение оператора **INSERT**.

5. Приведите примеры всех известных вам форматов оператора **INSERT**.

6. Объясните назначение оператора **SELECT**.

7. Объясните назначение следующих ключевых слов, применяемых в операторе SELECT: **DISTINCT, TOP, FROM, WHERE, BETWEEN, NOT, IS NULL, IN, ALL, ANY, AND, OR, LIKE, EXISTS, GROUP BY, CUBE, ROLLUP, HAVING, ORDER BY, COMPUTE**.

8. Поясните назначение и способ применения следующих агрегатных функций: **COUNT, SUM, MAX, MIN, AVG**.

9. Поясните назначение и принцип применения конструкции **SELECT ... INTO ...**.

10. Поясните назначение и принцип применения команд **UNION, UNION ALL, INTERSECT** и **EXCEPT**.

11. Объясните назначение оператора **DELETE**.

12. Приведите примеры всех известных форматов оператора **DELETE**.

13. Объясните назначение и принцип применения оператора **TRUNCATE** и поясните его отличие от оператора **DELETE**.

## Практическая работа № 5

# СОЗДАНИЕ И УДАЛЕНИЕ ПРЕДСТАВЛЕНИЙ

### 5.1. Теоретические сведения

#### 5.1.1. Общие сведения о представлениях

Представления – это объекты базы данных, представляющие собой именованный результат выполнения фиксированного **SELECT**-запроса. В секции **FROM** оператора **SELECT** представления могут использоваться точно так же как таблицы, поэтому часто представления называют виртуальными таблицами.

В некоторых случаях при работе с представлениями допускается применение операторов **INSERT**, **DELETE** и **UPDATE**, которые проецируются на одну соответствующую представлению таблицу.

Создать, удалить и изменить представление можно с помощью SMS или, как и любой другой объект базы данных, с помощью операторов **CREATE**, **DROP**, **ALTER** языка T-SQL.

#### 5.1.2. Создание и удаление представлений с помощью SMS

Для создания представления можно воспользоваться контекстным меню SMS (рис. 5.1), позволяющим создать вкладку (рис. 5.2), предназначенную для формирования **SELECT**-запроса.

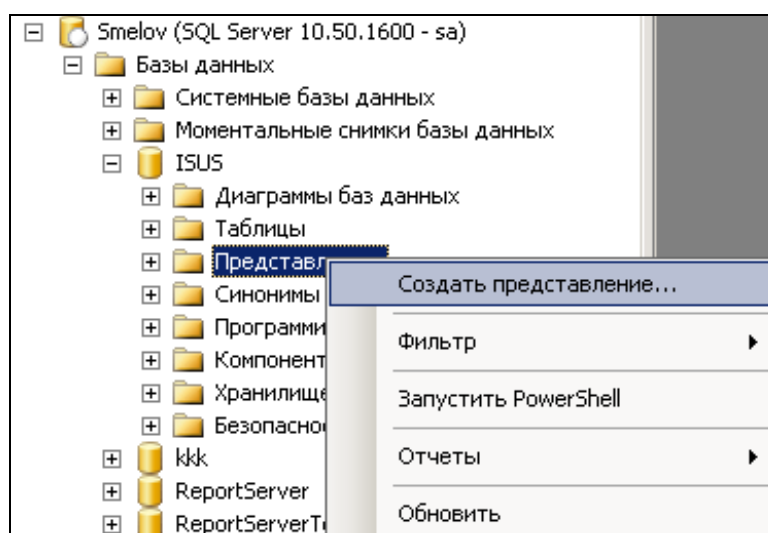


Рис. 5.1. Создание представления



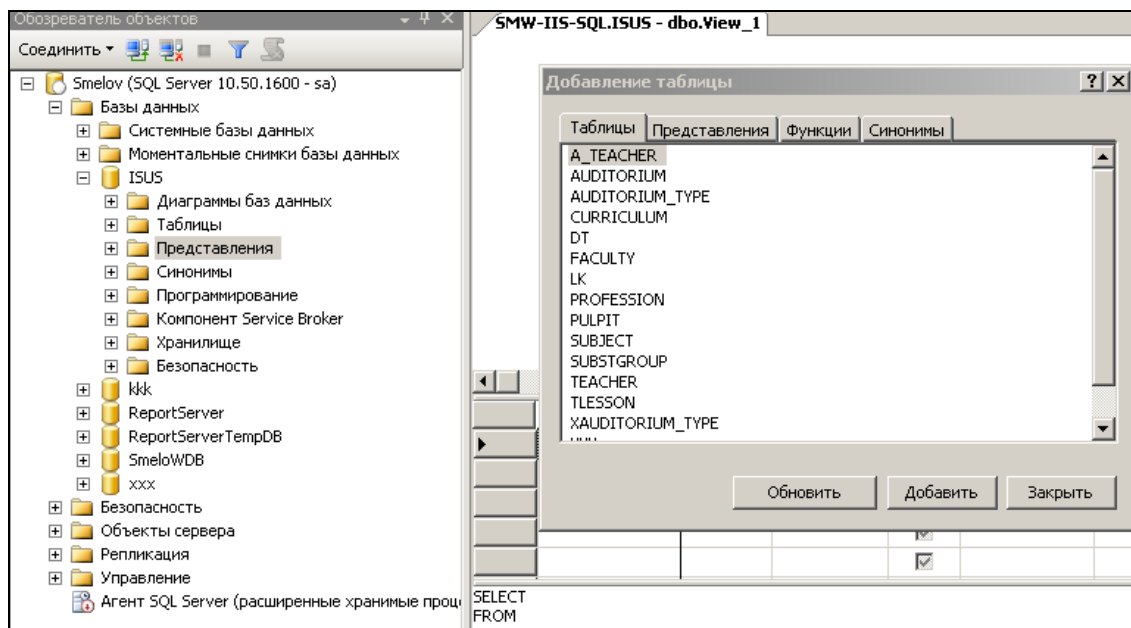


Рис. 5.2. Формирование SELECT-запроса

Для удаления существующего представления тоже можно использовать контекстное меню, установив предварительно курсор на удаляемое представление.

### 5.1.3. Создание и удаление представлений с помощью T-SQL

Создать представление можно с помощью SQL-оператора **CREATE VIEW**. Если представление уже существует, то сценарий для его создания может быть получен автоматически с помощью контекстного меню в SMS (рис. 5.3).

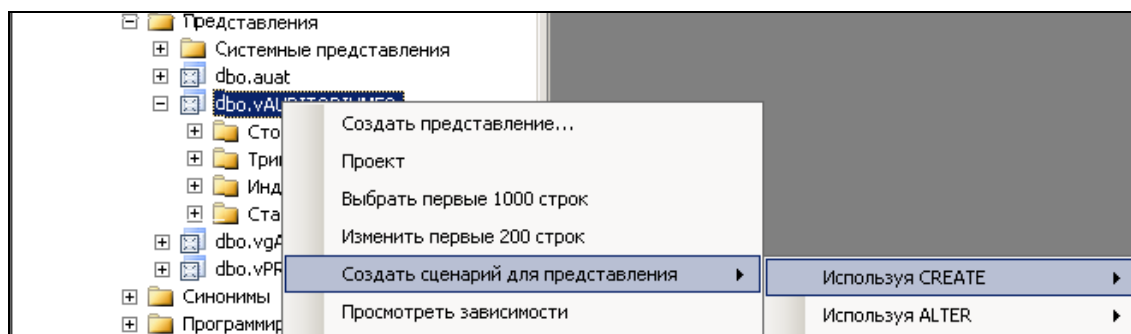


Рис. 5.3. Формирование сценария для создания представления

Аналогично может быть построен сценарий для удаления представления с помощью оператора **DROP** или для изменения с помощью **ALTER**.

При создании с помощью конструкции **WITH CHECK OPTION** для представления может быть указано его специальное свойство, которое позволяет рассматривать логическое выражение в **WHERE** как дополнительное ограничение, проявляющееся при выполнении операторов **INSERT** и **UPDATE**, применяемых к этому представлению.

С полным синтаксисом операторов **CREATE**, **DROP** и **ALTER VIEW** можно ознакомиться в [5].

## 5.2. Задания

### 5.2.1. Задание 18. Создание представлений с помощью SMS

1. Создайте представление с помощью SMS, которое отображает фамилию, имя и отчество всех преподавателей только кафедры **ИСиТ**; проверьте работоспособность представления с помощью оператора **SELECT**.

2. Создайте представление с помощью SMS, которое отображает все наименования факультетов и фамилию, имя и отчество преподавателей, работающих на этих факультетах.

3. Создайте представление с помощью SMS, которое отображает количество преподавателей, работающих на каждом факультете.

4. Сформируйте с помощью SMS сценарии для создания представлений, созданных в предшествующих пунктах задания.

### 5.2.2. Задание 19. Создание представлений с помощью T-SQL

1. Создайте представление с помощью T-SQL, которое отображает все лекционные аудитории и их вместимость.

2. Создайте представление с помощью T-SQL, которое отображает среднюю вместимость каждого типа аудиторий.

3. Разработайте самостоятельно представление со свойством **WITH CHECK OPTION** и допускающее выполнение операторов **INSERT** и **UPDATE**, продемонстрируйте его работоспособность и действие свойства **WITH CHECK OPTION**.

4. Разработайте самостоятельно представление, **SELECT**-запрос которого использует другое представление.

### 5.2.3. Задание 20. Применение представлений

1. Пр продемонстрируйте простейший **SELECT**-запрос к представлению, разработанному в предыдущих заданиях.

2. Разработайте SELECT-запрос к представлению.
3. Разработайте SELECT-запрос, содержащий секции **WHERE** и **ORDER BY**, к разработанному в предыдущих заданиях представлению.
4. Разработайте SELECT-запрос, содержащий секцию **GROUP BY**, к разработанному в предыдущих заданиях представлению.
5. Разработайте DELETE-запрос, содержащий подзапрос к разработанному в предыдущих заданиях представлению.
6. Разработайте UPDATE-запрос, содержащий подзапрос к разработанному в предыдущих заданиях представлению.

#### 5.2.4. Задание 21. Контрольные вопросы

1. Дайте определение представлению.
2. Поясните, в каких случаях целесообразно использовать представление.
3. Перечислите операторы SQL, с помощью которых представления создаются, удаляются и изменяются.
4. Перечислите способы создания представлений.
5. Перечислите существующие ограничения при создании представлений.
6. Поясните назначение и принцип действия секции **WITH CHECK OPTION**.
7. Перечислите правила создания представлений, допускающих выполнение операций **INSERT**, **DELETE**, **UPDATE**.

## **Практическая работа № 6**

# **СОЗДАНИЕ И УДАЛЕНИЕ ИНДЕКСОВ**

### **6.1. Теоретические сведения**

#### **6.1.1. Общие сведения об индексах**

Индекс – это объект базы данных, предназначенный для ускорения запросов к данным в таблице базы данных.

Создать, удалить или изменить индекс можно с помощью SMS или с помощью операторов **CREATE**, **DROP**, **ALTER** языка T-SQL.

При создании индекса указывается один или несколько столбцов таблицы, по значениям которых будет построен и поддерживаться индекс. Индекс представляет собой структуру памяти, организованную в виде сбалансированного дерева. В узлах дерева содержатся страницы со значениями из выбранных столбцов. SELECT-запросы, использующие в секции WHERE столбцы таблицы, для которых построен индекс, не требуют сканирования всей таблицы, так как индекс позволяет получить указатели на все запрашиваемые строки за небольшое количество операций чтения. Кроме того, индексы, как правило, имеют размеры значительно меньшие, чем таблицы, что в большинстве случаев позволяет их размещать в оперативной памяти сервера. Использование индексов является прозрачным для программиста, пишущего запросы. Он не имеет возможности указать серверу, какой индекс следует использовать в запросе. Вопрос о применении индекса для выборки данных решается специальной компонентой сервера, называемой *оптимизатор запросов*. Программист может лишь выяснить порядок выборки данных, получив *план исполнения запроса*.

При создании таблицы, содержащей столбцы, имеющие свойства **PRIMARY KEY** или **UNIQUE**, индексы создаются автоматически. Кроме того, индексы могут быть уникальными и не уникальными. Уникальный индекс действует для столбца или группы столбцов таблицы как ограничение целостности **UNIQUE**.

Следует отметить, что помимо «классических» типов индексов MSS поддерживает еще ряд специфических типов: XML-индексы, SPATIAL-индексы, FULLTEXT-индексы. Более подробно с возможностями, предоставляемыми этими индексами, можно ознакомиться в [4, 8]

### 6.1.2. Кластеризованные и некластеризованные индексы

MSS поддерживает два типа индексов: *кластеризованные* и *некластеризованные* индексы.

При создании кластеризованного индекса данные индексируемой таблицы располагаются в физическом порядке, соответствующем индексу, и становятся частью кластеризованного индекса. Поэтому кластеризованный индекс для таблицы может быть создан только один.

Некластеризованный индекс – это отдельный объект, имеющий указатели на строки таблицы. Максимальное количество некластеризованных индексов для одной таблицы не должно превышать 1000.

### 6.1.3. Создание и удаление индексов с помощью SSMS

Для создания индексов с помощью SSMS следует воспользоваться контекстным меню, как это показана на рис. 6.1.

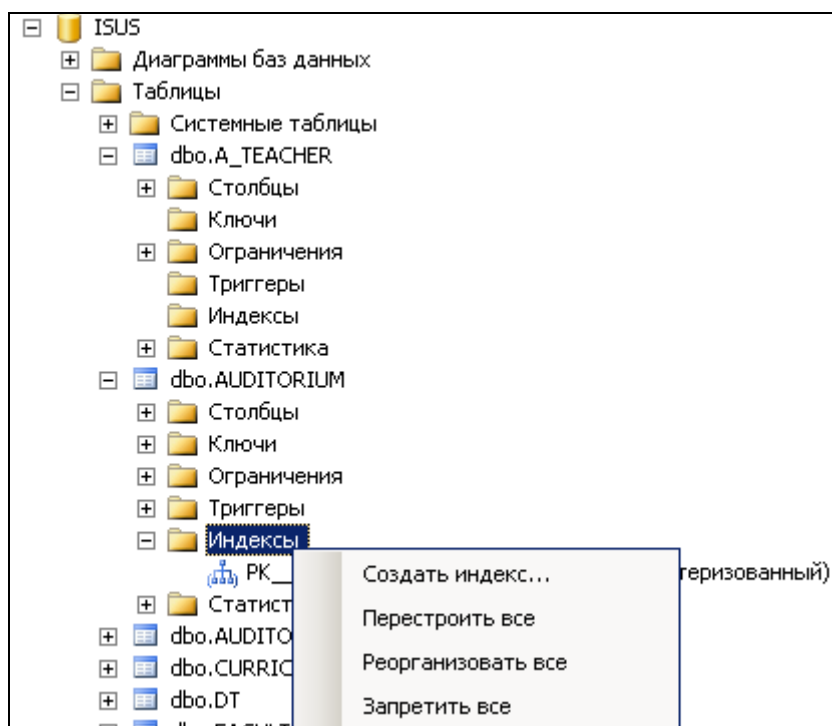


Рис. 6.1. Создание индекса

Удаление индекса осуществляется таким же образом.

### 6.1.4. Создание и удаление индексов с помощью T-SQL

Как и любой объект базы данных индекс может быть создан с помощью оператора **CREATE**, а удален с помощью оператора **DROP**.

Для существующего индекса с помощью SSMS достаточно просто можно получить готовые скрипты T-SQL для создания и удаления этого индекса. На рис. 6.2 представлен пример применения контекстного меню SSMS для формирования скрипта, создающего указанный индекс.

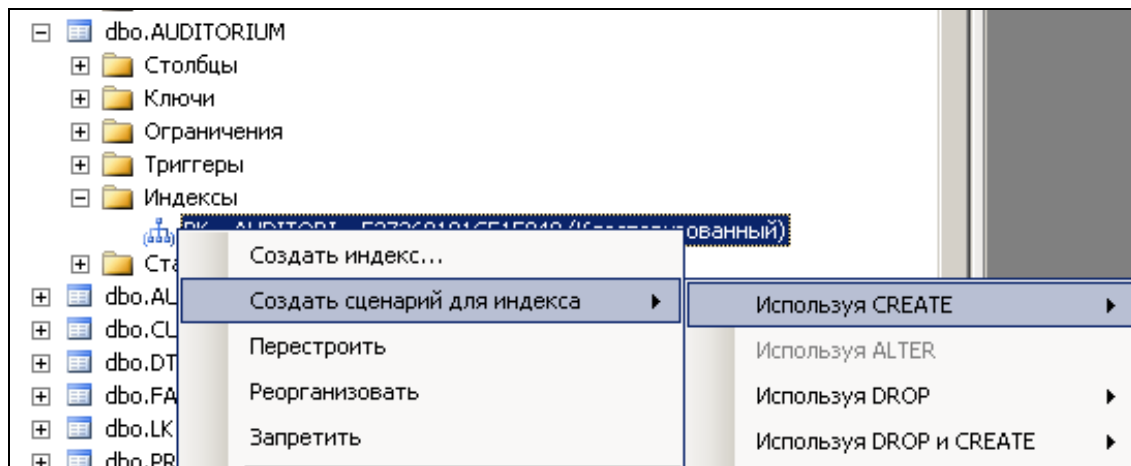


Рис. 6.2. Формирование скрипта для создания индекса

### 6.1.5. Перестроение и реорганизация индексов

Если в проиндексированной таблице осуществляется большое количество изменений (*INSERT*, *UPDATE*, *DELETE*), то на страницах индекса остаются неиспользованные фрагменты при полном заполнении самой страницы. Это явление называется **фрагментацией индексов**. Дело в том, что память, выделенная в странице индекса, не освобождается в оперативном режиме, так как это для сервера слишком затратная операция, которая может нивелировать весь эффект от применения индекса. Фрагментация приводит к тому, что индекс увеличивается в объеме и это в конечном итоге приводит к снижению производительности сервера.

Такая организация индексов приводит к тому, что для таблиц, информация в которых часто изменяется, требуется периодическая перестройка индексов.

Для перестройки индексов можно воспользоваться SSMS, выполнив пункты «Перестроить» или «Реорганизовать» в контекстном меню (рис. 6.2) или выполнив операторы.

Аналогичного эффекта можно добиться с помощью операторов ***ALTER INDEX ... REBUILD*** и ***ALTER INDEX ... REORGANIZE***.

Основное отличие перестроения индекса от его реорганизации заключается в том, что в первом случае индекс пересоздается полностью. Такого же эффекта можно достичь удалением (***DROP***) и созда-

нием (**CREATE**) этого индекса. Перестроение полностью уничтожает фрагментацию в индексе. В случае реорганизации осуществляется перестроение только нижнего (листового) уровня индекса, что позволяет в большинстве случаев существенно снизить фрагментацию. При этом время затрачивается на реорганизацию индекса значительно меньше, чем на его перестроение.

Перестроение индекса может выполняться в оперативном режиме. По умолчанию на время выполнения этой операции сервер заблокирует таблицу, ассоциированную с индексом, что приведет к тому, что таблица перестанет быть доступной пользователям. Для того чтобы таблица была доступна, следует в операторе **ALTER INDEX ... REBUILD** указать ключевое слово **ONLINE**.

Реорганизация индекса всегда выполняется в оперативном режиме и не вызывает долгосрочной блокировки таблицы.

## 6.2. Задания

### 6.2.1. Задание 22. Работа с индексами с помощью SSMS

1. С помощью SSMS определите, выпишите и укажите в отчете все индексы в базе данных ISUS, созданные автоматически.

2. Создайте с помощью SSMS некластеризованный уникальный индекс с именем IX\_UNIQUE\_TEACHER\_NAME для столбца TEACHER\_NAME таблицы TEACHER.

3. Продемонстрируйте, что создание индекса IX\_UNIQUE\_TEACHER\_NAME аналогично созданию ограничения целостности **UNIQUE** для столбца TEACHER\_NAME таблицы TEACHER.

4. Создайте с помощью SSMS некластеризованный уникальный индекс с именем IX\_SUBJECT\_NAME для столбца SUBJECT\_NAME таблицы SUBJECT.

5. С помощью SSMS выполните реорганизацию индекса IX\_UNIQUE\_TEACHER\_NAME.

6. С помощью SSMS выполните перестроение индекса IX\_SUBJECT\_NAME.

7. С помощью SSMS удалите индекс IX\_SUBJECT\_NAME.

### 6.2.2. Задание 23. Работа с индексами с помощью T-SQL

1. С помощью SSMS создайте скрипт для удаления индекса IX\_UNIQUE\_TEACHER\_NAME.

2. С помощью SSMS создайте скрипт для создания индекса IX\_UNIQUE\_TEACHER\_NAME.

3. Последовательно выполните скрипты на удаление и создание индекса IX\_UNIQUE\_TEACHER\_NAME, убедитесь, что он создан.

4. Разработайте скрипт на T-SQL, создающий индекс IX\_SUBJECT\_NAME, аналогичный создаваемому в предыдущем задании.

5. Разработайте скрипт на T-SQL, реорганизуя индекс IX\_UNIQUE\_TEACHER\_NAME и перестраивающий индекс IX\_UNIQUE\_TEACHER\_NAME.

### 6.2.3. Задание 24. Контрольные вопросы

1. Дайте определение индексу и поясните его назначение.
2. Назовите все типы индексов, поддерживаемые MSS.
3. Чем отличается кластеризованный индекс от некластеризованного?
4. В каких случаях индексы создаются автоматически?
5. Что такое фрагментация индексов, в каких случаях она возникает?
6. Чем отличаются процедуры перестроения и реорганизации индексов?
7. Для чего применяется ключевое слово **ONLINE** в операторе перестроения индекса?



## Практическая работа № 7

### ОСНОВЫ T-SQL

#### 7.1. Теоретические сведения

##### 7.1.1. Структура программы на T-SQL

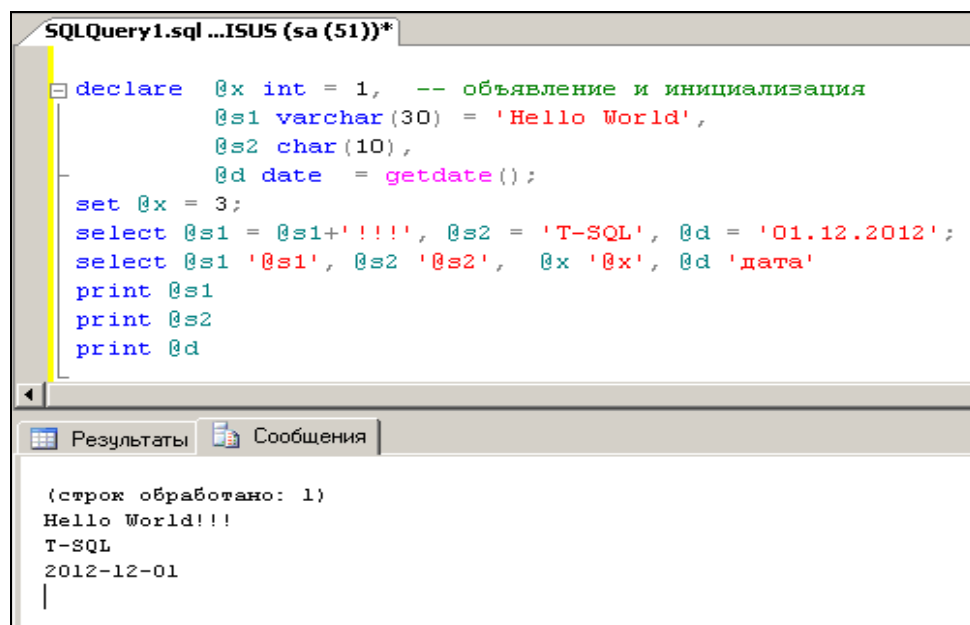
Язык T-SQL представляет собой расширение языка SQL и предназначен для разработки программ, исполняемых под управлением MSS.

В простейшем случае программа на T-SQL может состоять из одного оператора. В примерах к практической работе № 4 рассматривались DML-операторы, каждый из которых можно назвать программой на T-SQL.

Более сложные программы могут содержать объявление и инициализацию переменных, операторы присвоения, ветвления, циклов и т. п. Код программ может быть записан в виде обыкновенной последовательности операторов, а может быть организован в виде процедур или функций, позволяющих этот код использовать повторно.

##### 7.1.2. Команды T-SQL

Переменные, используемые в программах на T-SQL, бывают двух типов: локальные и глобальные. Локальные переменные служат для хранения промежуточных результатов, их имена начинаются с символа **@**.



The screenshot shows a SQL Server Enterprise Manager window titled "SQLQuery1.sql ...ISUS (sa (51))\*". The script contains the following T-SQL code:

```
declare @x int = 1, -- объявление и инициализация
@s1 varchar(30) = 'Hello World',
@s2 char(10),
@d date = getdate();
set @x = 3;
select @s1 = @s1+'!!!', @s2 = 'T-SQL', @d = '01.12.2012';
select @s1 '@s1', @s2 '@s2', @x '@x', @d 'дата'
print @s1
print @s2
print @d
```

The results pane at the bottom shows the output of the script:

```
(строк обработано: 1)
Hello World!!!
T-SQL
2012-12-01
|
```

Рис. 7.1. Вывод результатов в выходной поток

Глобальные переменные – это predefined переменные. Они предназначены для хранения информации сервера, их имена начинаются с @@. Операторы разделяются символом ; ( точка с запятой).

Объявление переменных выполняется с помощью команды **DECLARE**, присвоение значений – с помощью команд **SET** или **SELECT**.

Вывод значений переменных в выходной поток можно осуществить с помощью команды **PRINT**, а в результирующий набор – с помощью команды **SELECT** (рис. 7.1, 7.2).

Результаты		Сообщения		
	@s1	@s2	@x	дата
1	Hello World!!!	T-SQL	3	2012-12-01

Рис. 7.2. Вывод в результирующий набор

Команды **IF ... ELSE** и **CASE** позволяют проверить логическое условие (рис. 7.3).

SQLQuery2.sql ...ISUS (sa (55))\*

SQLQuery1.sql ...ISUS (sa (51))\*

```

declare @x int = 1, -- объявление и инициализация
@s1 varchar(30) = 'Hello World',
@s2 char(10),
@d date = getdate();

if @x > 0
    select '0x > 0'
else
    select '0x <= 0'

select case
    when @x > 10 then '0x > 10'
    when @x > 5 then '0x > 5'
    else '0x < 5'
end

```

Результаты

Сообщения

(Отсутствует имя столбца)

1 @x > 0

(Отсутствует имя столбца)

1 @x > 5

Рис. 7.3. Вывод в результирующий набор

Команда **WHILE** позволяет организовать цикл. В качестве операторных скобок используется конструкция **BEGIN ... END** (рис. 7.4).

```
declare @x int = 1,  
        @n int = 3  
while @n > 0  
begin  
    print @x;  
    set @x = @x+1;  
    set @n = @n-1  
end;
```

Рис. 7.4. Команда WHILE и операторные скобки

Для обработки ошибок используется конструкция **TRY ... CATCH**. Программист может сам сгенерировать ошибку с помощью команды **RAISEERROR** (рис. 7.5).

```
declare @x int = 1,  
        @n int = 3  
begin try  
    set @n = @x/0; -- деление на 0  
end try  
begin catch  
    if error_number() = 8134  
        raiserror ('деление на 0', 16, 1);  
    else  
        print 'неизвестная ошибка, код ' + error_number();  
end catch;
```

Рис. 7.5. Обработка ошибок

Для ознакомления с более полным набором команд T-SQL и применением операторов DML в программах T-SQL рекомендуется [9].

### 7.1.3. Встроенные функции

В состав T-SQL входит множество различных встроенных функций, существенно облегчающих программирование (рис. 7.6).

Все функции разбиваются на следующие категории: математические, строковые, для работы с датами, системные функции и функции конфигурирования.

```

print '---- примеры математических функций'
print PI() -- число ПИ
print power(2,3) -- возведение в степень
print round(2.12274567,3) -- округление
print '---- примеры строковых функций'
print ltrim(' 12345'); -- удалить пробелы слева
print rtrim('12345 '); -- удалить пробелы справа
print substring('12345', 2,2); -- подстрока
print '---- примеры функций для работы с датами'
print getdate() -- текущая дата и время
print day(getdate()) -- текущий день месяца
print month(getdate()) -- текущий месяц года
print '---- примеры системных функций'
print cast(1 as varchar(3)); -- преобразование типа
print host_name(); -- имя хоста
print current_user; -- имя пользователя
print '---- функции конфигурирования'
print @@version; -- версия сервера
print @@servername; -- имя сервера
print @@spid; -- регистрационный номер сеанса

```

Рис. 7.6. Примеры встроенных функций

## 7.2. Задания

### 7.2.1. Задание 25. Разработка программы на T-SQL

Разработайте программу, итеративно уменьшающую вместимость всех аудиторий на 5% за итерацию до тех пор, пока суммарная вместимость не станет равной 1000 или меньше.

### 7.2.2. Задание 26. Применение встроенных функций

1. Приведите примеры семи математических функций, не представленных на рис. 7.6.
2. Приведите примеры семи строковых функций, не представленных на рис. 7.6.
3. Приведите примеры семи функций для работы с датами и временем, не представленных на рис. 7.6.
4. Приведите примеры семи системных функций, не представленных на рис. 7.6.
5. Приведите примеры семи функций конфигурирования, не представленных на рис. 7.6.

### **7.2.3. Задание 27. Контрольные вопросы**

1. Определите правила именования переменных в T-SQL.
2. Перечислите типы данных, которые могут использоваться для переменных T-SQL.
3. Назовите конструкцию T-SQL, с помощью которой организуются операторные скобки.
4. Назовите команду объявления и инициализации переменных T-SQL.
5. Назовите команды присвоения значений переменным T-SQL.
6. Назовите команды T-SQL для вывода информации в выходной поток и результирующий набор.
7. Назовите команды T-SQL для логического сравнения значений.
8. Назовите команды T-SQL для организации цикла.
9. Назовите команды T-SQL перехода.
10. Назовите команду T-SQL для вывода сообщения об ошибке.
11. Назовите конструкцию T-SQL для обработки ошибок.
12. Поясните, что значит «встроенная функция».
13. Перечислите семь встроенных математических функций и поясните их назначение.
14. Перечислите семь встроенных строковых функций и поясните их назначение.
15. Перечислите семь встроенных функций для работы с датами и поясните их назначение.
16. Перечислите семь встроенных системных функций и поясните их назначение.
17. Перечислите семь встроенных функций конфигурирования и поясните их назначение.

## Практическая работа № 8

# ПРИМЕНЕНИЕ КУРСОРОВ

### 8.1. Теоретические сведения

#### 8.1.1. Понятие курсора

Курсор – это механизм, позволяющий в программе на языке T-SQL обрабатывать отдельные строки, полученные в результате SELECT-запроса. В основе любого курсора лежит SELECT-запрос, который описывает набор строк, полученный в результате исполнения этого запроса. Иногда курсор представляют как указатель на область памяти (буфер), в который полностью или частично помещены строки SELECT-запроса.

Курсоры бывают трех типов: динамические, статические и ключевые.

Динамические курсоры – курсоры, в которых каждая считанная строка имеет последнюю версию на ее момент считывания.

Статические курсоры – курсоры, изолированные от текущего состояния базы данных. Все строки ассоциированного SELECT-запроса считываются один раз в момент процедуры открытия курсора.

Ключевые курсоры – курсоры, занимающие промежуточное положение между статическими и динамическими.

Кроме того, курсоры различаются по области видимости (локальные и глобальные), по операциям с данными (только для чтения и для обновления), по порядку чтения строк (только вперед и с произвольных позиционированием) и по другим параметрам. Более подробно механизм курсоров описывается в [4, 9].

#### 8.1.2. Общая схема применения курсора

Прежде всего, курсор необходимо объявить. Объявление курсора осуществляется в команде **DECLARE**. Объявление содержит имя курсора, SELECT-запрос, описывающий набор строк, а также набор параметров курсора.

Затем курсор следует открыть. Открытие курсора осуществляется с помощью команды **OPEN**, в которой указывается имя открываемого курсора. Действия сервера, происходящие при открытии курсора, зависят от типа этого курсора.

После открытия курсора может быть осуществлена выборка строк с помощью команды **FETCH**. Каждое выполнение команды **FETCH**

позволяет считать в переменные содержимое столбцов одной строки. Возможности (и соответствующие параметры) команды ***FETCH*** зависят от параметров курсора. В простейшем случае ***FETCH*** считывает следующую строку в том порядке, в котором это определено в ассоциированном с курсором SELECT-запросе. Результат выполнения команды ***FETCH*** отражается в виде содержимого глобальной переменной @@***FETCH\_STATUS***.

Как правило, выполнение команды ***FETCH*** осуществляется в цикле с проверкой переменной @@***FETCH\_STATUS*** на каждой итерации.

При завершении работы с курсором выполняется команда ***CLOSE***, позволяющая освободить ресурсы сервера, обеспечивающие поддержку курсора.

Для глобальных курсоров следует еще выполнить команду ***DEALLOCATE***. Если этого не сделать, то сервер будет «помнить» объявление этого курсора и цикл работы с ним может быть повторен, начиная с команды ***OPEN***.

На рисунке приведен текст программы на языке T-SQL, использующей глобальный, статический курсор, применяемый только для чтения.

```
use ISUS
GO
declare teacher_curs cursor global static read_only
    for select TEACHER, TEACHER_NAME from TEACHER;
declare @t varchar(20), @tn varchar(200);
open teacher_curs;
fetch next from teacher_curs into @t, @tn;
while @@FETCH_STATUS = 0
begin
    print 'код = ' + @t + 'наименование = ' + @tn
    fetch next from teacher_curs into @t, @tn;
end;
close teacher_curs;
deallocate teacher_curs;
```

Пример программы на T-SQL, использующей курсор

Если курсор предназначен для изменения или удаления строк, то обычно в таких случаях применяются операторы ***UPDATE*** и ***DELETE*** с конструкцией ***WHERE CURRENT OF***, которая позволяет изменить или удалить текущую строку курсора. Более подробно о работе с курсорами можно ознакомиться в [4, 9].

## 8.2. Задания

### 8.2.1. Задание 28. Работа с курсорами

1. Разработайте программу, использующую локальный, динамический курсор для считывания всех строк таблицы AUDITORIUM и увеличения значений в столбце AUDITORIUM\_CAPACITY на 5% с применением конструкции **UPDATE ... WHERE CURRENT OF**.

2. Приведите примеры применения динамического и статического курсоров, демонстрирующих их отличие.

### 8.2.2. Задание 29. Работа со SCROLL-курсорами

Разработайте программу на языке T-SQL, применяющую курсор со свойством **SCROLL**, и продемонстрируйте все возможности позиционирования в операторе **FETCH**, которые допускаются для таких курсоров.

### 8.2.3. Задание 30. Контрольные вопросы

1. Дайте определение курсору и поясните его назначение.
2. Перечислите все типы курсоров и объясните их особенности.
3. Перечислите все свойства курсоров, которые можно задать при их объявлении.
4. Перечислите порядок действий при работе с курсором.
5. Перечислите все разновидности позиционирования оператором **FETCH** для курсоров со свойством **SCROLL**.



## Практическая работа № 9

# ПРИМЕНЕНИЕ ПРОЦЕДУР И ФУНКЦИЙ

### 9.1. Теоретические сведения

#### 9.1.1. Хранимые процедуры, их создание, удаление и применение

Программы на языке T-SQL могут быть достаточно объемными. Кроме того, часто одни и те же фрагменты кода удобно использовать в одной или разных программах. Для структуризации кода, а также для хранения повторно используемого кода в языке T-SQL применяются процедуры и функции.

Процедура – это объект базы данных, представляющий собой поименованный фрагмент кода T-SQL, который можно исполнять в программе T-SQL с помощью команды **EXECUTE**. Процедура может иметь параметры, значение которых устанавливает вызывающая программа. Параметры могут быть входные и выходные. При вызове процедуры параметры могут задаваться в позиционном или параметрическом виде. Процедура может возвращать числовое значение типа **int**, в этом случае процедура должна заканчиваться командой **RETURN**. Кроме того, процедура может формировать результирующий набор, обрабатываемый в вызывающем коде. Процедура создается с помощью оператора **CREATE**, удаляется с помощью оператора **DROP**, а также может быть изменена с помощью оператора **ALTER**. Следует отметить, что при создании процедуры могут быть указаны дополнительные свойства, с полным перечнем которых можно ознакомиться в [4, 9].

На рис. 9.1 продемонстрировано создание простейшей процедуры, имеющей два параметра, а на рис. 9.2 – вызов этой процедуры двумя способами: с позиционной и параметрической передачей параметров.

```
create procedure CFAC @s varchar(20), @c int output
as begin
    set @c = (select COUNT(*) from FACULTY where upper (FACULTY_NAME) like upper ('%'+@s+'%'));
    if @c > 0 select FACULTY from FACULTY where upper (FACULTY_NAME) like upper ('%'+@s+'%');
    return 1;
end;
```

Рис. 9.1. Пример создания процедуры

```

declare @rc int, @cc int;
execute @rc = CFAC 'лес', @cc output
print @rc;
print @cc;
execute @rc = CFAC @c = @cc out, @s = 'лес'
print @rc;
print @cc;

```

Рис. 9.2. Вызовы процедуры

### 9.1.2. Функции, их создание, удаление и применение

Функция – это объект базы данных, представляющий собой поименованный фрагмент кода T-SQL, который можно использовать в выражениях языка T-SQL.

Главное отличие функций от процедур заключается в том, что функция может возвращать любые за малым исключением типы значений. В функции запрещено выполнение любых изменений в базе данных, параметры могут быть только входные, а при вызове используется только позиционный формат передачи данных; кроме того, функция не может формировать результирующий набор, но может возвращать результат SELECT-запроса или сформированную в памяти таблицу.

Различают три вида пользовательских функций: скалярные (возвращают единственное значение), табличные (возвращают результат SELECT-запроса) и табличные многооператорные (возвращают сформированную в памяти таблицу). Функция создается с помощью оператора **CREATE**, удаляется с помощью оператора **DROP**, а также может быть изменена с помощью оператора **ALTER**. Следует отметить, что при создании функции могут быть указаны дополнительные свойства, с полным перечнем которых можно ознакомиться в [4, 9].

На рис. 9.3 продемонстрировано создание простейшей табличной функции, имеющей один параметр, а на рис. 9.4 – вызов этой функции.

```

create function FFAC(@s varchar(20)) returns table
as return select FACULTY
        from FACULTY
        where upper (FACULTY_NAME) like upper('%'+@s+'%');

```

Рис. 9.3. Пример создания табличной функции

```
select * from FFAC('лес')
```

Рис. 9.4. Вызов табличной функции

## 9.2. Задания

### 9.2.1. Задание 31. Создание и удаление хранимых процедур

1. Разработайте хранимую процедуру, принимающую два параметра (один входной и один выходной), формирующую код возврата, значение выходного параметра, а также результирующий набор.

2. Продемонстрируйте вызов разработанной процедуры с позиционной передачей параметров.

3. Продемонстрируйте вызов процедуры с параметрической передачей параметров.

4. Удалите процедуру.

### 9.2.2. Задание 32. Создание и удаление функций

1. Разработайте скалярную функцию без параметров.

2. Разработайте табличную функцию с одним параметром.

3. Разработайте многооператорную табличную функцию с двумя параметрами.

4. Разработайте программу на T-SQL, вызывающую разработанные функции.

### 9.2.3. Задание 33. Контрольные вопросы

1. Дайте определение хранимой процедуре.

2. Какой тип значений может возвращать процедура?

3. Какие виды параметров могут использоваться в процедуре?

4. Какие способы передачи параметров допускаются при вызове процедуры?

5. Дайте определение пользовательской функции.

6. Перечислите отличия функций от хранимых процедур.

7. Перечислите виды функций и поясните их особенности.

## Практическая работа № 10

# ТРИГГЕРЫ И ИХ ПРИМЕНЕНИЕ

### 10.1. Теоретические сведения

#### 10.1.1. Понятие триггера, типы триггеров

Триггер – это особый вид хранимых процедур, вызываемых сервером при возникновении определенных событий в базе данных. Различают два вида триггеров: системные и DML-триггеры.

Системные триггеры предназначены для реагирования на события сервера и здесь не рассматриваются.

Каждый DML-триггер закрепляется за определенной таблицей или представлением и вызывается на события, связанные с добавлением (*INSERT*), удалением (*DELETE*) или изменением (*UPDATE*) строк таблиц. MSS поддерживает два вида триггеров: *AFTER* и *INSTEAD OF*.

К одной таблице допускается прикрепление нескольких триггеров.

Триггеры *AFTER* срабатывают после наступления события в ассоциированной таблице или представлении.

Триггеры *INSTEAD OF* срабатывают вместо события, а само событие никогда не наступает.

При исполнении триггера MSS обеспечивает создание и заполнение двух таблиц *INSERTED* и *DELETED*, доступных в коде триггера. Таблицы позволяют получить удаленные, измененные (до изменения и после), а также добавленные строки в соответствующем событии.

Более подробно о триггерах можно прочитать в [4, 9].

#### 10.1.2. Создание и удаление триггеров

Создание триггеров осуществляется с помощью оператора *CREATE*, удаление – с помощью оператора *DROP*.

На рис. 10.1 приведен пример создания AFTER-триггера, который по наличию строк в таблицах *INSERTED* и *DELETED* определяет тип обрабатываемого события.

На рис. 10.2 приведен пример создания INSTEAD OF-триггера, аналогичный по своему назначению триггеру, представленному на рис. 10.1.

```

create trigger TR_SUBJECT on dbo.SUBJECT
for insert, update, delete
as
    if exists (select * from inserted) and
        exists (select * from deleted)
        print 'Update';
    else if exists (select * from inserted) and
        not exists (select * from deleted)
        print 'Insert';
    else if not exists (select * from inserted) and
        exists (select * from deleted)
        print 'Delete';

```

Рис. 10.1. Пример создания AFTER-триггера

```

create trigger ITR_SUBJECT on dbo.SUBJECT
instead of insert, update, delete
as
    if exists (select * from inserted) and
        exists (select * from deleted)
        print 'Update';
    else if exists (select * from inserted) and
        not exists (select * from deleted)
        print 'Insert';
    else if not exists (select * from inserted) and
        exists (select * from deleted)
        print 'Delete';
    else print 'Unknown';

```

Рис. 10.2. Пример создания INSTEAD OF-триггера

## 10.2. Задания

### 10.2.1. Задание 34. Применение AFTER-триггеров

1. Повторите пример триггера на рис. 10.1.
2. Добавьте, измените и удалите строку в таблице SUBJECT и объясните результат.
3. Измените триггер так, чтобы в нем выводилось содержимое непустых таблиц **INSERTED** и **DELETED**.
4. Добавьте, измените и удалите строку в таблице SUBJECT и объясните результат.

### 10.2.2. Задание 35. Применение INSTEAD OF-триггеров

1. Повторите пример триггера на рис. 10.2.
2. Добавьте, измените и удалите строку в таблице SUBJECT и объясните результат.
3. Измените триггер так, чтобы в нем выводилось содержимое непустых таблиц ***INSERTED*** и ***DELETED***.
4. Добавьте, измените и удалите строку в таблице SUBJECT и объясните результат.

### 10.2.3. Задание 36. Контрольные вопросы

1. Дайте определение триггеру.
2. Перечислите типы триггеров, поддерживаемых MSS.
3. Перечислите виды DDL-триггеров, поддерживаемых MSS.
4. Поясните разницу между видами DDL-триггеров.
5. Поясните назначение и принцип заполнения таблиц ***INSERTED*** и ***DELETED***.

## ЛИТЕРАТУРА

1. Microsoft Download Center [Электронный ресурс] / Компания Microsoft. – М., 2012. – Режим доступа: <http://www.microsoft.com/downloads>. – Дата доступа: 12.09.2012.
2. Microsoft SQL Server // Википедия [Электронный ресурс] / Wikimedia Foundation Inc. – М., 2012. – Режим доступа: [http://ru.wikipedia.org/wiki/Microsoft\\_SQL\\_Server](http://ru.wikipedia.org/wiki/Microsoft_SQL_Server). – Дата доступа: 12.09.2012.
3. Microsoft SQL Server 2008: руководство администратора для профессионалов / Б. Найт [и др.]. – М.: Вильямс, 2010. – 944 с.
4. ITBand [Электронный ресурс]. – М., 2012. – Режим доступа: <http://itband.ru/2010/07/install-microsoft-sql-server-2008-r2/>. – Дата доступа: 12.09.2012.
5. Библиотека MSDN [Электронный ресурс] / Компания Microsoft. – М., 2012. – Режим доступа: <http://msdn.microsoft.com/ru-ru/library/>. – Дата доступа: 12.09.2012.
6. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт. – М.: Вильямс, 2005. – 1316 с.
7. Сайт БГТУ [Электронный ресурс] / БГТУ. – Минск, 2012. – Режим доступа: <http://www.bstu.unibel.by>. – Дата доступа: 12.09.2012.
8. Хотек, М. Microsoft SQL Server 2008: Реализация и обслуживание. Учебный курс Microsoft / М. Хотек. – М.: Русская редакция, 2012. – 576 с.
9. Жилинский, А. А. Самоучитель Microsoft SQL Server / А. А. Жилинский. – СПб.: БХВ-Петербург, 2009. – 240 с.

# ОГЛАВЛЕНИЕ

<b>Предисловие .....</b>	<b>3</b>
<b>Введение .....</b>	<b>5</b>
<b><i>Практическая работа № 1. Установка программного обеспечения .....</i></b>	<b>7</b>
1.1. Теоретические сведения .....	7
1.2. Задания .....	11
<b><i>Практическая работа № 2. Создание и удаление базы данных ...</i></b>	<b>13</b>
2.1. Теоретические сведения .....	13
2.2. Задания .....	16
<b><i>Практическая работа № 3. Создание и удаление таблиц .....</i></b>	<b>18</b>
3.1. Теоретические сведения .....	18
3.2. Задания .....	20
<b><i>Практическая работа № 4. Применение SQL DML.....</i></b>	<b>24</b>
4.1. Теоретические сведения .....	24
4.2. Задания .....	27
<b><i>Практическая работа № 5. Создание и удаление представлений.....</i></b>	<b>32</b>
5.1. Теоретические сведения .....	32
5.2. Задания .....	34
<b><i>Практическая работа № 6. Создание и удаление индексов .....</i></b>	<b>36</b>
6.1. Теоретические сведения .....	36
6.2. Задания .....	39
<b><i>Практическая работа № 7. Основы T-SQL .....</i></b>	<b>41</b>
7.1. Теоретические сведения .....	41
7.2. Задания .....	44
<b><i>Практическая работа № 8. Применение курсоров .....</i></b>	<b>46</b>
8.1. Теоретические сведения .....	46
8.2. Задания .....	48
<b><i>Практическая работа № 9. Применение процедур и функций ....</i></b>	<b>49</b>
9.1. Теоретические сведения .....	49
9.2. Задания .....	51
<b><i>Практическая работа № 10. Триггеры и их применение .....</i></b>	<b>52</b>
10.1. Теоретические сведения .....	52
10.2. Задания .....	53
<b>Литература.....</b>	<b>55</b>



## **БАЗЫ ДАННЫХ**

Составитель **Смелов** Владимир Владиславович

Редактор *О. П. Приходько*

Компьютерная верстка *О. П. Приходько*

Корректор *О. П. Приходько*

Издатель:

УО «Белорусский государственный технологический университет».

ЛИ № 02330/0549423 от 08.04.2009.

Ул. Свердлова, 13а, 220006, г. Минск.